USERS

¡EL LENGUAJE QUE REEMPLAZARÁ A FLASH YA ESTÁ AQUÍ!

HTML

ENTIENDA EL CAMBIO, APROVECHE SU POTENCIAL

por Damián De Luca

LA EVOLUCIÓN DE HTML: NUEVAS CARACTERÍSTICAS / MULTIMEDIA SIN NECESIDAD DE FLASH
DISEÑOS CON ESTILOS CSS3 / TRANSFORMACIONES, TRANSICIONES Y ANIMACIONES
COMPATIBILIDAD CON NAVEGADORES Y MÓVILES / CARACTERÍSTICAS AVANZADAS

SERS MANUALES USERS MANUALES USERS MANUALES USERS MANUALES USERS

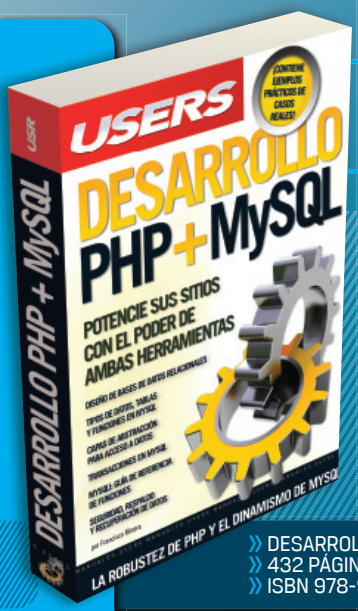
APLIQUE EL NUEVO ESTÁNDAR A SUS PROYECTOS

CONÉCTESE
CON LOS MEJORES LIBROS DE COMPUTACIÓN

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

usershop.redusers.com
usershop@redusers.com

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

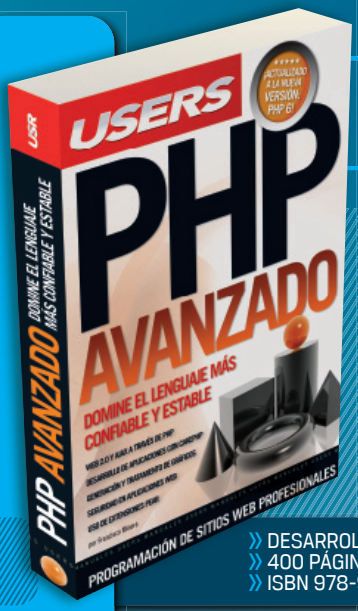


POTENCIE SUS SITIOS CON EL PODER DE AMBAS HERRAMIENTAS

» DESARROLLO / PHP
» 432 PÁGINAS
» ISBN 978-987-1773-16-9

CLAVES Y TÉCNICAS PARA OPTIMIZAR SITIOS DE FORMA PROFESIONAL

» DESARROLLO / INTERNET
» 400 PÁGINAS
» ISBN 978-987-1773-09-1



PROGRAMACIÓN DE SITIOS WEB PROFESIONALES

» DESARROLLO / PHP
» 400 PÁGINAS
» ISBN 978-987-1773-07-7

APRENDA CÓMO CREAR BLOGS ATRACTIVOS SIN CONOCIMIENTOS PREVIOS

» 200 RESPUESTAS
» 320 PÁGINAS
» ISBN 978-987-663-037-5

HTML5

ENTIENDA EL CAMBIO, APROVECHE SU POTENCIAL

por Damián De Luca

RedUSERS

TÍTULO: HTML5

AUTOR: Damián de Luca

COLECCIÓN: Manuales USERS

FORMATO: 17 x 24 cm

PÁGINAS: 320

Copyright © MMXI. Es una publicación de Fox Andina en coedición con Gradi S.A. Hecho el depósito que marca la ley 11723. Todos los derechos reservados. Esta publicación no puede ser reproducida ni en todo ni en parte, por ningún medio actual o futuro sin el permiso previo y por escrito de Fox Andina S.A. Su infracción está penada por las leyes 11723 y 25446. La editorial no asume responsabilidad alguna por cualquier consecuencia derivada de la fabricación, funcionamiento y/o utilización de los servicios y productos que se describen y/o analizan. Todas las marcas mencionadas en este libro son propiedad exclusiva de sus respectivos dueños. Impreso en Argentina. Libro de edición argentina. Primera impresión realizada en Sevagraf, Costa Rica 5226, Grand Bourg, Malvinas Argentinas, Pcia. de Buenos Aires en VIII, MMXI.

ISBN 978-987-1773-79-4

De Luca, Damián

HTML5. - 1a ed. - Buenos Aires : Fox Andina; Dalaga, 2011.

320 p. ; 24x17 cm. - (Manual users; 216)

ISBN 978-987-1773-79-4

1. Informática. I. Título

CDD 005.3

ANTES DE COMPRAR

EN NUESTRO SITIO PUEDE OBTENER, DE FORMA GRATUITA, UN CAPÍTULO DE CADA UNO DE LOS LIBROS EN VERSIÓN PDF Y PREVIEW DIGITAL. ADEMÁS, PODRÁ ACCEDER AL SUMARIO COMPLETO, LIBRO DE UN VISTAZO, IMÁGENES AMPLIADAS DE TAPA Y CONTRATAPA Y MATERIAL ADICIONAL.

RedUSERS
COMUNIDAD DE TECNOLOGIA

redusers.com

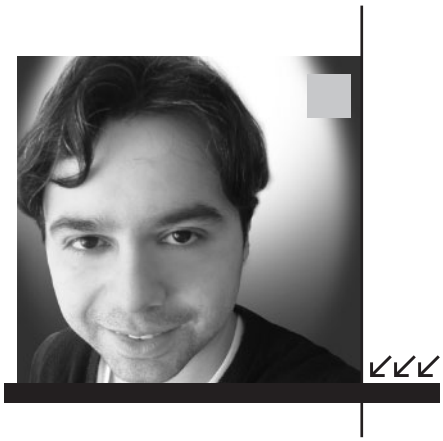
Nuestros libros incluyen guías visuales, explicaciones paso a paso, recuadros complementarios, ejercicios, glosarios, atajos de teclado y todos los elementos necesarios para asegurar un aprendizaje exitoso y estar conectado con el mundo de la tecnología.

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com // usershop@redusers.com

Damián de Luca



Nació en la ciudad de Buenos Aires, Argentina, en 1977. Ha realizado estudios sobre electrónica y lenguajes de programación.

Es autor de los libros 101 Secretos de Vista, Hardware Extremo y Webmaster Profesional. Se ha desempeñado como editor de libros relacionados con informática publicados por esta editorial.

Ha colaborado como editor en la revista Users Digital Design, publicación para la cual también ha escrito notas sobre diseño y desarrollo web.

Además, participa como autor en colecciones de fascículos y contenidos online relacionados con informática.

Es consultor, docente y desarrollador web, con amplios conocimientos en XHTML, CSS, AJAX, PHP y Bases de datos. Es especialista en CSS3 y HTML5.

E-mail: contacto@damiandeluca.com.ar
Web: http://damiandeluca.com.ar
Twitter: @damiande

Agradecimientos

A todos los que confiaron en mí para este proyecto y a todos aquellos que, desde su lugar, hicieron que este libro fuera posible. También quiero agradecer especialmente a los que, a lo largo de mi vida, me ayudaron a nutrirme de conocimientos para llegar a ser hoy quien les escribe estas páginas.

Dedicatorias

Dedicado especialmente a mi hijo Gael y a mi mujer Myriam por ser quienes me escuchan, me aman y me sostienen cada día. Esta obra también está dedicada a todos mis afectos, para aquellos que me quieren de verdad y siempre están a mi lado.

Prólogo

Decir que HTML5 ya está aquí es una realidad; también lo es expresar la idea de que ha causado aún más interés del que se esperaba. Sin duda HTML5 representa un avance en pos de una Web más amigable para los desarrolladores y usuarios. Esto se demuestra en el interés del público por las novedades que nos llegan de la mano de la versión 5 de HTML y en el trabajo dedicado por los desarrolladores a la tarea de dar soporte a este estándar.

La localización geográfica y caché para aplicaciones son algunas de las novedades de HTML5. Se trata de funciones que permiten a los desarrolladores de contenido web determinar en forma bastante certera la ubicación del usuario, pudiendo desplegar información relacionada con ella. Por otro lado, gracias a la caché de aplicaciones, tendremos acceso a la posibilidad de guardar y utilizar aplicaciones online cuando no existe una conexión a Internet. Además, la posibilidad de realizar el procesamiento en segundo plano aprovechando la capacidad de la PC es una realidad mientras usamos un navegador web.Por otro lado HTML5 incorpora la capacidad de embeber videos y de que estos puedan ser reproducidos sin un plugin adicional.

Como ya sabemos, HTML5 es el futuro y está entre nosotros. Como usuarios, solo nos queda disfrutar de todas las características que nos ofrece; como desarrolladores tendremos que aprender a implementarlas y, para ayudarnos a transitar este camino, nada mejor que apoyarnos en la experiencia y los conocimientos de Damián de Luca, quien ha logrado plasmar esta obra eminentemente práctica y con todo lo que necesitamos para dominar HTML5. ¡Que lo disfruten!

Claudio Peña
Autor y editor RedUsers
prologo@redusers.com

El libro de un vistazo

Este libro está enfocado en brindarle al lector las posibilidades que introduce HTML5. Veremos la introducción a HTML5 para develar sus características principales, cómo se lo puede aplicar en la práctica, y cuáles son los lenguajes y tecnologías asociadas útiles para nuestros desarrollos.

*01 TECNOLOGÍAS Y ESTÁNDARES WEB

En el primer capítulo de esta obra nos encargaremos de realizar una breve recorrida que nos permitirá conocer la evolución que han experimentado los estándares más importantes que se emplean para el desarrollo y diseño de sitios web. Para continuar nos daremos a la tarea de analizar también el progreso en las técnicas, las principales herramientas de desarrollo que tenemos a nuestra disposición y el papel de los navegadores en esta historia.

*02 NOVEDADES DE HTML5

En la primera parte de este capítulo, conoceremos cada uno de los elementos que nos ofrecen tanto HTML4 como XHTML. Para continuar, haremos un análisis que nos mostrará la forma en que evolucionan sus características con la llegada de la versión 5 de HTML. También nos encargaremos de conocer cuáles son los elementos que proceden a retirarse del estándar, los que reciben algún tipo de cambio y, por supuesto, las novedades que se incorporan a partir de HTML5, de esta forma estaremos preparados para enfrentar el desarrollo teniendo presentes las nuevas opciones.

*03 ELEMENTOS ESTRUCTURALES Y SEMÁNTICA

En este capítulo vamos a profundizar sobre la importancia que cobra la semántica en el lenguaje HTML5. A través de las secciones que lo componen conoceremos y analizaremos cómo se debe pensar la estructura de los documentos cuando usamos HTML5 y posteriormente vamos a aprender de qué manera podemos utilizar correctamente los elementos destinados a dicho fin. Además, nos introduciremos en temas importantes como Microdatos.

*04 TRABAJO CON ESTILOS

A través de este importante capítulo veremos cómo interactúa HTML5 con las hojas de estilo correspondientes a CSS 2.1 y también CSS3 para acceder a representar los contenidos en el medio de salida que consideremos adecuado. Nos encargaremos de analizatr y conocer algunos de los principales conceptos realcionados, como selectores, clases y propiedades, también aprenderemos a construir reglas de estilos. Por último recorreremos las propiedades principales que nos llegan con CSS 2.1 y CSS3, y veremos los valores que les podremos aplicar a cada una de ellas.

*05 MULTIMEDIA

Conoceremos las principales características multimedia que ofrece HTML5. Analizaremos las posibilidades que nos brindan y de qué manera podemos utilizar las etiquetas de audio y video, que se incorporan a partir de esta versión del lenguaje. Además, nos encargaremos de profundizar sobre los aspectos que nos ayudarán a aprender cómo aplicarlas en nuestros proyectos web.

*06 FORMULARIOS

Conoceremos los aspectos relacionados con la creación de formularios web. Comenzaremos analizando las posibilidades que nos ofrecen HTML4 y XHTML para la creación de formularios, luego veremos cómo se puede potenciar el desarrollo empleando JavaScript y AJAX. Para completar, profundizaremos sobre las características más importantes que se incorporan a partir de HTML5.

*07 DISEÑO Y DESARROLLO WEB ADAPTADO A MÓVILES

En este capítulo veremos aspectos vinculados con el desarrollo de soluciones web para móviles. Conoceremos las técnicas de detección de características y pantallas de dispositivos móviles. Conoceremos las ventajas que introduce HTML5, y analizaremos las herramientas y los frameworks que nos ayudarán en el desarrollo y testeo.

*08 CARACTERÍSTICAS AVANZADAS

En el capítulo final analizaremos las características avanzadas que introducen HTML5 y las APIs relacionadas. Entre los temas que abordaremos se destaca el acceso al DOM, Canvas, SVG, WebGL, WebSockets, Geolocalización, Drag&Drop nativo, WebStorage, Web Messaging, Server-Sent Events, Web Workers y la posibilidad de creación de aplicaciones offline con Application cache API, entre otras.

INFORMACIÓN COMPLEMENTARIA

A lo largo de este manual podrá encontrar una serie de recuadros que le brindarán información complementaria: curiosidades, trucos, ideas y consejos sobre los temas tratados. Para que pueda identificarlos en forma más sencilla, cada recuadro está identificado con diferentes iconos:

CURIOSIDADES E IDEAS

ATENCIÓN

DATOS ÚTILES Y NOVEDADES

SITIOS WEB

Contenido

Sobre el autor 4
Prologo 5
El libro en un vistazo 6
Informacion complementaria 7
Inroduccion 12

*01

Tecnologías y estándares web

Los orígenes de la Web 14
Web 2.0 14
Web 3.0 15
W3C y la importancia de la estandarización 17
Lenguajes de etiquetas 18
SGML 19
XML 19
HTML 20
XHTML 23
HTML5 25
Otros lenguajes y tecnologías relacionados con el desarrollo web 26
CSS 26
CSS3 30
JavaScript 32
AJAX 34
Tecnologías y lenguajes del lado servidor 41
Navegadores 41
Internet Explorer 42
Mozilla Firefox 44
Google Chrome 45
Safari 47
Opera 48
Los motores de renderizado 51
Herramientas de desarrollo 53
Software WYSIWYG 53
Otras alternativas 55
Inspección de código desde el navegador 57
Test de compatibilidad 59
Técnicas de detección de compatibilidad 60
Resumen 61
Actividades 62

*02

Novedades de HTML5

Qué son los elementos HTML 64
Los atributos en HTML 64
Los eventos en HTML 66
Elementos HTML4/XHTML 68
Elementos que se retiran del estándar o cambian con HTML5 86
Atributos que dejan de ser soportados 87
Elementos que cambian con HTML5 88
Características que incorpora HTML5 89
Atributos en HTML5 91
Los eventos que se incorporan con HTML5 92
Resumen 97
Actividades 98

*03

Elementos estructurales y semántica

Semántica 100
Declaración de página y cabecera del documento 102
Estructura semántica de documentos en HTML5 104
Atributos soportados 108
Estructurar una página en HTML5 110
Adaptar la estructura semántica 111

Pensar la estructura de páginas HTML5 desde cero 113
Compatibilidad 119
Microdatos 120
SEO 122
Resumen 125
Actividades 126

*04
Trabajo con estilos

Interacción de HTML5 con los estilos CSS 128
Selectores, clases, y propiedades 129
Construcción de reglas en CSS 131
Propiedades shorthand 133
Herencia, cascada y especificidad 134
Herencia 134
Cascada y especificidad 136
Estilos con CSS 2.1 137
Selectores avanzados 150
Pseudoelementos y pseudoclases 151
Unidades de medida para el desarrollo web 153
CSS3: nuevas características 155
Sombra en el texto 156
Multicolumna 159
@fontface 160
Otras propiedades relacionadas con texto 162
Sombreado en la caja 163
Bordes 164
Fondos 167
Color 168
Diseños flexibles con el módulo Flexible Box Layout 170
Propiedades de interfaz de usuario 172
Selectores, pseudoelementos y pseudoclases en CS3 173
Transformación 176
Transición 180
Opciones de animación 182
Resumen 183
Actividades 184

*05
Multimedia

Evolución del concepto multimedia en la Web 186
Incorporar audio con HTML5 191
Códecs de audio 192
Incorporar video con HTML5 193
La etiqueta <video> 194
Códecs de video 195
El elemento track 197
La etiqueta <source> 198
Aspectos de compatibilidad de audio y video 199
Incorporar los elementos multimedia utilizando HTML5 en nuestro desarrollo 202
Potenciar y personalizar nuestro desarrollo con frameworks de JavaScript 205
Acceso a dispositivos multimedia 208
Resumen 209
Actividades 210

*06
Formularios

Evolución en el uso de los formularios en la Web 212
Formularios HTML/XHTML 213
La etiqueta <form> 213
La etiqueta <input> 215
La etiqueta <label> 218
La etiqueta <textarea> 219
Las etiquetas <fi eldset> y <legend> 220
Las etiquetas <select>, <option> y <optgroup> 221

La etiqueta <button>.....222
JavaScript y AJAX para potenciar los formularios.....223
Características del formulario que incorpora HTML5 226
Nuevos atributos para <form>..... 226
La nueva etiqueta <datalist> 226
La nueva etiqueta <ouput> 227
Las novedades en <input>.....227
Los cambios para <textarea>.....231
Otros cambios para etiquetas de formulario.....232
Incorporar características HTML5 en un formulario.....233
Crear un formulario HTML5 desde cero..... 238
Resumen.....239
Actividades.....240

*07

Diseño y desarrollo web adaptado a móviles

La evolución de la Web móvil 242
Ventajas de HTML5 para las plataformas móviles 244
Navegadores para móviles 245
Mobile Safari 245
Navegador Android 246
Internet Explorer Mobile..... 247
Firefox para móviles 248
Opera Mini y Opera Mobile 248
Otros navegadores para móviles 249
Cómo saber si el móvil está preparado para HTML5 251
Técnicas de detección.....251
Detección del dispositivo con CSS 252
CSS3 Media Queries.....254
Herramientas de desarrollo web para móviles 256
Crear aplicaciones web.....258
Frameworks para desarrollo de soluciones para móviles 259
Resumen.....260
Actividades.....261

*08

Características avanzadas

JavaScript y DOM.....264
API de selectores 264
Seleccionar elementos del DOM..... 265
Canvas 266
SVG.....274
WebGL (3D).....277
WebSockets.....279
Geolocalización.....281
Detección de compatibilidad 284
Geolocalización con Google Maps 285
Drag & Drop 288
Web Storage.....290
Almacenamiento local en nuestros desarrollos 291
Web Messaging.....293
Server-Sent Events.....295
Web Workers 296
Application Cache API 298
Cache Manifest..... 299
History API.....300
Resumen.....301
Actividades.....302

*

Servicios al lector

Índice temático 304
Sitios web relacionados..... 307
Programas relacioados 311

Introducción

La evolución de las tecnologías y los lenguajes vinculados con Internet tienen un período de trabajo importante, hasta que llegan a ser un estándar recomendado por la entidad que los impulsa. El **W3C** (*World Wide Web Consortium*) es el consorcio que trabaja sobre las especificaciones de los estándares de Internet.

HTML5 llega en una etapa en la cual la Web vive importantes cambios, un tiempo en el que se necesita una renovación en los estándares vigentes para poder ser potenciados y abrir nuevas puertas en el desarrollo web.

La estandarización de una nueva versión de un lenguaje requiere que los documentos que componen su especificación recorran un camino, en el cual serán tratados por equipos de trabajo especializados y analizados por comités de expertos.

Para comprender mejor el tema debemos saber que, hasta llegar a ser recomendación, los documentos de trabajo que componen una especificación pasan por diversas etapas y períodos en los que son revisados.

La primera de las etapas es conocida como **Working Draft** (borrador de trabajo); en ella, se publica el primer documento de trabajo y se realizan las primeras revisiones y debates sobre el documento. Dentro de la etapa de borrador, cuando el documento logra madurez, puede llegar a la instancia conocida como **Last Call** (última llamada) donde se verifica si el material se encuentra apto para seguir su camino a la siguiente etapa.

El documento luego pasa a **Candidate Recommendation** (recomendación candidata); aquí se puede identificar si el documento está apto para su implementación o si, por el contrario, debe regresar a la etapa anterior. En la siguiente etapa, denominada **Proposed Recommendation** (recomendación propuesta), el documento es revisado por

un comité asesor. Finalmente, cuando se cumplen de manera satisfactoria todas las instancias, el documento llega a ser **W3C Recommendation** (recomendación del W3C). Cada documento debe pasar por todas las etapas para llegar a ser recomendación del W3C.

Al momento de escribir este libro, HTML5 se encuentra en la etapa de Working Draft, y entrará en el período de Last Call a partir de mayo de 2011. Según lo informado por el W3C, HTML5 llegará a ser recomendación en el año 2014.

La pregunta que se pueden hacer algunos lectores es: "Si HTML5 aún no es un estándar, ¿por qué debo aprender a usarlo ahora?". La respuesta es clara: porque esta nueva versión del lenguaje de etiquetas sobre el que se basa la Web ya comenzó a ser utilizada por los desarrolladores de todo el mundo, y los principales navegadores del mercado (tanto Desktop como Mobile) han comenzado a brindar soporte para varias de sus características, ofreciéndoles a los usuarios la posibilidad de comenzar a disfrutar las ventajas que introduce HTML5.

Como desarrolladores web, nos vemos inmersos en un mercado muy competitivo, que nos conduce a la necesidad de estar informados y actualizados de manera constante. Esta necesidad nos sitúa en una carrera en la que no podemos quedar atrás.

En conclusión, el futuro es hoy, HTML5 ya está entre nosotros. Este libro representa la posibilidad de introducirnos en un tema que, acompañado por una serie de tecnologías, está cambiando el diseño y el desarrollo web tal como lo conocíamos. Bienvenidos a esta nueva etapa de la Web.

Damián de Luca

*01 –

Tecnologías y estándares web

Para comenzar a comprender la llegada de HTML5, en el primer capítulo de este libro vamos a realizar una breve recorrida que nos permitirá conocer la evolución que han experimentado los estándares utilizados para el desarrollo y diseño de sitios web. Analizaremos también el progreso en las técnicas, las herramientas de desarrollo y el importante papel que tienen los navegadores en este proceso evolutivo.

- Web 2.0 14
- Web 3.0 15
- W3C y la importancia de la estandarización 17
- Lenguajes de etiquetas..... 18
- Navegadores..... 41
- Los motores de renderizado.... 51
- Herramientas de desarrollo 53
- Técnicas de detección de compatibilidad 60
- Resumen..... 61
- Actividades..... 62

Los orígenes de la Web

Internet no solo ha marcado uno de los más importantes avances tecnológicos del siglo XX, sino que también ha acompañado un cambio cultural de trascendencia que, en pleno siglo XXI, se mantiene en constante evolución. Pero toda historia tiene un comienzo, e Internet también lo tuvo, mucho antes de ser un fenómeno masivo.

La historia cuenta que el antecesor de Internet fue el proyecto conocido como **ARPANET**, una red descentralizada que algunos organismos estadounidenses utilizaron a partir de la década del sesenta. Sin embargo, el gran cambio se produciría entre fines de los ochenta y principios de los noventa, con la llegada de lo que se conoce como **World Wide Web**, es decir **WWW**, el sistema que se encarga de permitir la distribución de información mediante hipertexto.

De la mano de este cambio, comienza a popularizarse Internet en la población. Los usuarios ahora podían acceder a contenidos de la gran red, tan solo con disponer de una conexión mediante un módem y un navegador con la capacidad de interpretar contenidos de hipertexto. Esta etapa de Internet, que comprende aproximadamente desde principios de los noventa hasta el año 2003, es considerada como Web **1.0**.

El concepto de este primer paradigma de la Web responde a la idea de una web "estática" o de una "sola vía", donde el usuario es solo un "espectador" que recibe o lee contenidos, publicados por el **Webmaster** o dueño del sitio. Este paradigma se modificaría de manera sustancial con la llegada de la denominada Web 2.0.

Web 2.0

Los cambios en la Web no solo responden a temas tecnológicos, sino que estos van de la mano con la evolución de los hábitos de los usuarios, las tendencias en los modos de navegación, las necesidades del mercado y hasta con aspectos culturales que también influyen en este conjunto.

La **Web 2.0** representa principalmente un cambio cultural en Internet. Los usuarios, cansados de un rol pasivo, comienzan a buscar alternativas de participación. Nace una web social, donde los blogs, las redes sociales y las aplicaciones online son las estrellas.

Esto ocurre a partir del año 2004, con la aparición de técnicas y tecnologías como **AJAX** y la necesidad de los usuarios de expresarse de una manera mucho más fluida en la gran red de redes.

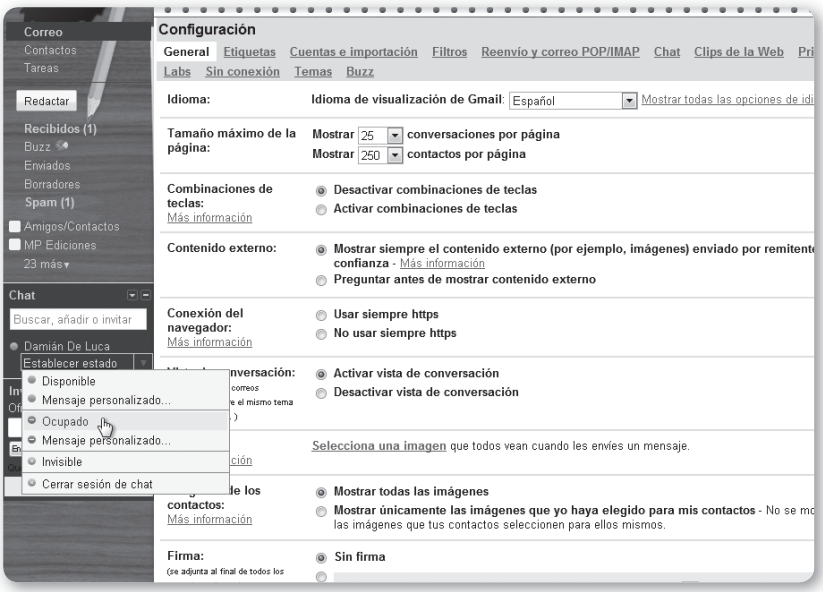


Figura 1. `Gmail` es, sin dudas, una de las herramientas que mejor representan a la Web 2.0. Se trata de una aplicación web con todas las características y la potencia de los programas de escritorio.

Web 3.0

El concepto de Web 3.0 es, quizás, más complejo de definir y discutido que el caso de sus predecesores: la Web 1.0 y 2.0. Existen diversas características que la definen, entre las cuales podemos mencionar: semántica, geolocalización, Web 3D, accesibilidad desde diversos dispositivos y también inteligencia artificial.

La Web **semántica**, como muchas veces se define a la Web 3.0, se refiere al uso de etiquetas o bien de metadatos para otorgar un significado semántico a los elementos de la Web. Esto posibilita cierta automatización

y la posibilidad de utilizar, con un mayor nivel de eficiencia, los agentes inteligentes que pueden realizar detección de contenidos.

Las características de **geolocalización**, muy empleadas en los equipos móviles, también han llegado a nuestro escritorio. Aunque aún pueden no ser tan precisas, las técnicas cada vez son más depuradas, y las mejoras en este campo no detienen su avance. Poder identificar a una persona, un dispositivo o cualquier elemento de manera geoespacial abre todo un mundo de posibilidades en el campo de la informática y, en especial, para todo lo referente a **Realidad Aumentada**.

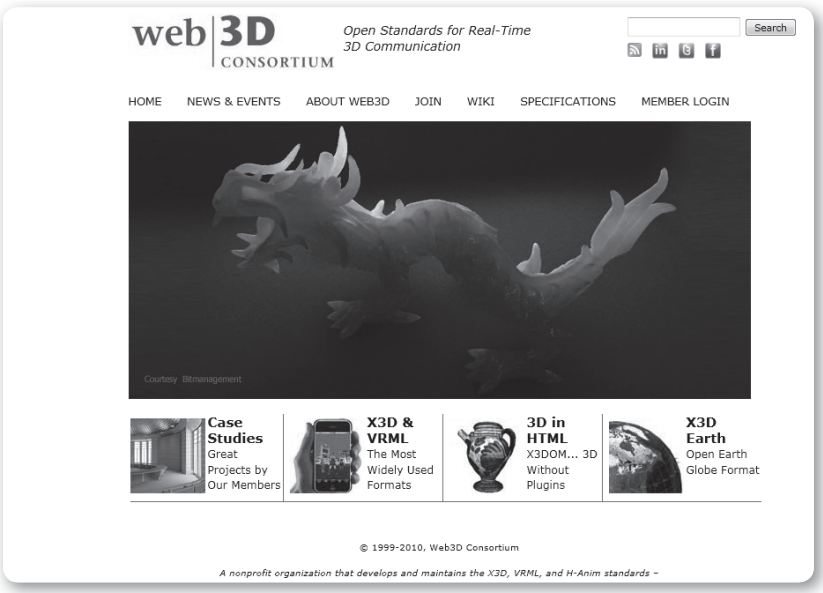


Figura 2. `Web3D Consortium` (`www.web3d.org`) es un organismo que tiene como objeto la definición de los estándares para la Web 3D.

Por otra parte, así como el cine renació con la aparición del **3D**, la Web también busca un nuevo horizonte con la presentación de contenidos en escenarios tridimensionales, y esto pretende ser una de las renovaciones que plantea la denominada Web 3.0.

La posibilidad de acceder desde distintos dispositivos es una realidad para una gran cantidad de usuarios y un desafío muy importante para diseñadores y desarrolladores web. Los usuarios ya no

están limitados a utilizar Internet desde una computadora de escritorio, ni siquiera dependen de una laptop. Teléfonos móviles, tablets, lectores de libros electrónicos y consolas de videojuegos son solo algunas de las posibilidades que se presentan para que el usuario pueda acceder a Internet en cualquier momento y desde cualquier lugar.

La **inteligencia artificial** es, seguramente, el ítem más ambicioso y también más difícil de alcanzar para la Web 3.0. Este concepto, muchas veces visto en producciones de ciencia ficción, puede comenzar a ser viable mediante algunos avances tecnológicos. Sin embargo, esta alternativa todavía no tiene la madurez necesaria para manejos prácticos, por lo cual aún se ubica en una etapa experimental.

W3C y la importancia de la estandarización

Definir modelos y normas es algo muy importante para lograr estándares que nos hagan a todos la vida más fácil en Internet.

Debemos tener en cuenta que desde los inicios de la Web, los problemas de compatibilidad causan dolores de cabeza a diseñadores y desarrolladores de todo el mundo; por tal motivo, el respeto a los estándares es un aspecto cada vez más valorado en este ámbito.

El **World Wide Web Consortium** (**W3C**) es el ente o consorcio, de alcance internacional, que se encarga de crear las reglas que se utilizan como recomendaciones fundamentales para la estandarización de los principales lenguajes y tecnologías utilizados en Internet, como el caso de HTML, CSS, XML, DOM y SVG, por ejemplo.

FACEBOOK: EL ÉXITO 2.0

Lanzada en el año 2004, la red social **Facebook** es uno de los exponentes de la Web 2.0 en lo que se refiere a redes sociales. Millones de personas en todo el mundo utilizan este servicio, que logra reunir a usuarios que pueden conectarse con sus familiares y amigos, pero que también ofrece la posibilidad de utilizar juegos y aplicaciones en línea creados por distintos desarrolladores.

El W3C surgió en el año 1994 y, bajo su órbita, hoy en día está el desarrollo de la versión 5 de HTML y el nivel 3 de CSS. Su sitio web oficial es **www.w3.org**. Es interesante mencionar que su director es nada menos que el "padre" de la Web: **Tim Berners-Lee**.

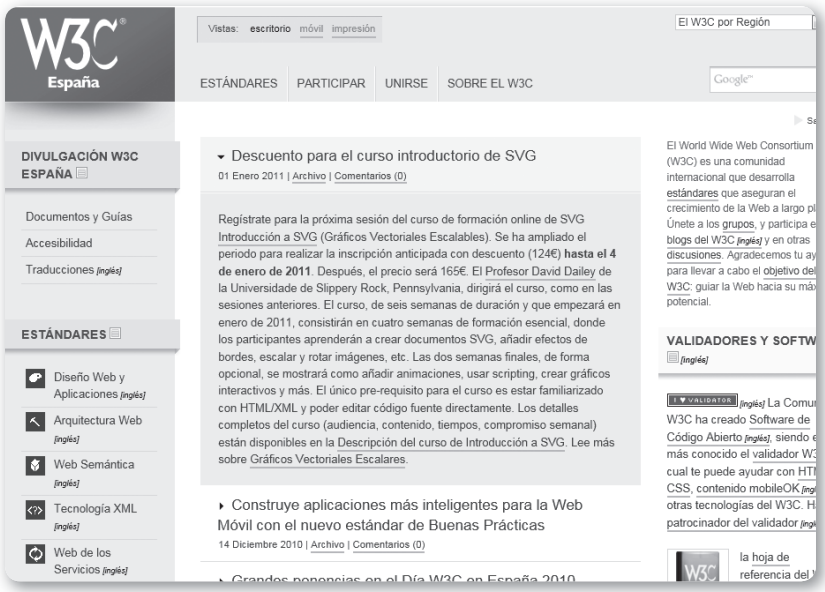


Figura 3. El W3C también se encarga de ofrecer información en idioma español; la encontramos en el sitio web **`http://w3c.es`**.

Lenguajes de etiquetas

Los **lenguajes de etiquetas**, también conocidos como lenguajes de marcado o de marcas, son los que nos permiten estructurar un documento mediante el uso de etiquetas. Un ejemplo muy popular de un lenguaje de etiquetas, conocido con seguridad por todos los que están leyendo este libro, es HTML.

Los lenguajes de etiquetas no se identifican con los de programación; esto ocurre principalmente porque los lenguajes de etiquetas no definen algunos aspectos básicos presentes en los

lenguajes de programación, como es el caso de funciones aritméticas o el uso de variables, por citar algunos ejemplos.

Los lenguajes de marcado tienen una historia bastante extensa. Si bien su concepto fue definido en la década del sesenta, sería en los setenta cuando surgirían las primeras aproximaciones a estos lenguajes y, en los ochenta, cuando llegarían las estandarizaciones. Claro está que recién en la década del noventa se harían masivos de la mano de HTML y gracias al enorme éxito de Internet.

SGML

Standard Generalized Markup Language, o simplemente conocido como **SGML**, es un estándar ISO que permite definir lenguajes de etiquetas. Es estándar ISO desde 1986 (ISO 8879:1986).

Una de las características más valoradas de SGML es la de permitir el intercambio de información de una manera sencilla, abstrayéndose de la complejidad de la aplicación. Otro aspecto para destacar es que permite definir nuevos lenguajes de marcado independientes.

Las etiquetas tienen la función de otorgarles nombres a los elementos, posibilitando marcar comienzo y final de un objeto lógico.

Entre los derivados de SGML, uno de los más destacados y conocidos es XML. También es importante mencionar que HTML se apoyó en SGML en su definición, sin basarse en el marcado estricto, tema que luego sería uno de los motivos de la llegada de XHTML (basado en XML).

XML

Bajo las siglas **XML** se conoce al metalenguaje de marcas extensible, cuyo nombre en inglés es *Extensible Markup Language*. Es importante aclarar que XML no es un lenguaje en sí mismo, sino que lo que nos ofrece es una manera de especificar reglas para definir lenguajes de etiquetas.

En la actualidad, es un estándar que está bajo la órbita del W3C, y su aplicación en Internet resulta frecuente.

XML es muy utilizado para intercambio de información, y su uso se ha visto potenciado con la llegada de AJAX. Este estándar permite describir datos de manera tal que se pueda realizar un intercambio de una forma transparente entre aplicaciones. Un caso de uso de XML que se puede mencionar es el del popular formato RSS, que se emplea para

compartir y difundir información en Internet. A continuación, podemos acceder a un claro jemplo de XML:

```
<?xml version="1.0"?>
<libro>
        <titulo>Webmaster Profesional</titulo>
        <autor>Damián De Luca</autor>
        <formato>Papel</formato>
        <peso>420 gr.</peso>
        <categoria>Desarrollo Web</categoria>
</libro>
```

Encontramos información sobre XML en **www.w3.org/TR/REC-xml**.

HTML

HTML (*HyperText Markup Language* o lenguaje de marcado de hipertexto) es el lenguaje de etiquetas que funciona como una de las piedras angulares de la World Wide Web. Aunque la evolución de Internet nos ha traído muchos avances en lo que se refiere a tecnología (Web 2.0 y Web 3.0, mediantes), el lenguaje de etiquetas que se popularizó en la década del noventa sigue siendo fundamental para el desarrollo web, ya que es el que comprenden e interpretan los navegadores. Claro está que por sí solo ya no es tan potente como lo fue en aquellos tiempos y, hoy por hoy, necesita combinarse con otras tecnologías y lenguajes para lograr resultados que estén a la altura de las necesidades del desarrollo web actual. Pero, para comprender un poco mejor la importancia de HTML, bien vale un pequeño repaso por su historia y sus características principales.

TIM BERNERS-LEE DEFINIÓ LA PRIMERA ESPECIFICACIÓN DE HTML A INICIOS DE LOS NOVENTA.

Breve historia

El lenguaje HTML aparece en los primeros años de la década del noventa y, como vimos, desciende de SGML. Por otro lado, en el año 1993, se logran dar algunos pasos muy importantes, como la

incorporación de etiquetas para imágenes, tablas y formularios. En 1995, llegaría HTML 2.0, que sería la primera versión en convertirse en estándar. En 1996, la estandarización de HTML pasa a manos del W3C y, en 1997, llega la recomendación conocida como HTML 3.2.

Entre los años 1997 y 1998, surge HTML 4, un paso importante en esta evolución. Luego llegaría el momento de HTML en su versión 4.01, que es publicado como una recomendación de la W3C a fines del año 1999 y que se trata del estándar que sige vigente hasta nuestros días.

Como veremos más adelante, XHTML llegaría para solucionar el tema de marcas estrictas y parecía ser el camino definitivo que debía seguirse; sin embargo, la historia cambió con la aparición de HTML5.

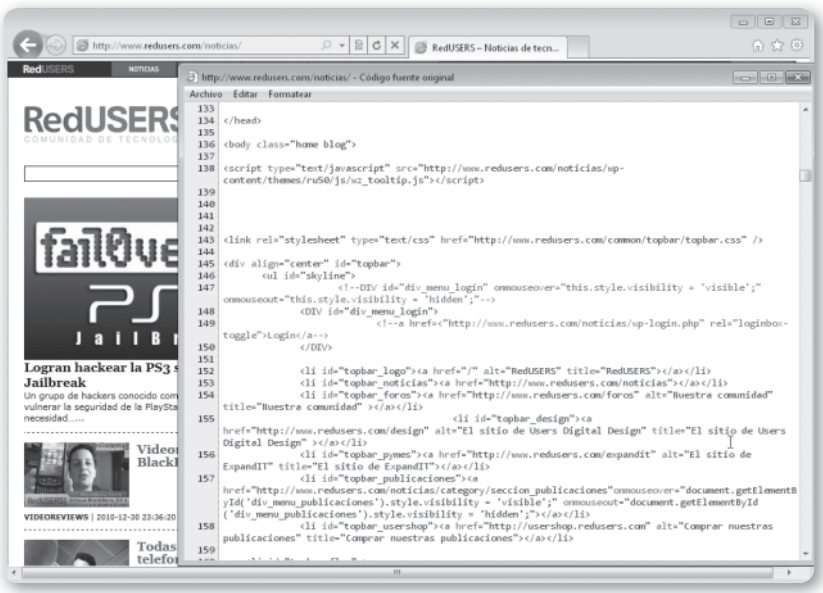


Figura 4. Los navegadores web permiten mostrar el código HTML, donde podremos identificar los diversos elementos que lo componen.

Características

Entre las características más importantes de HTML, destacamos que los elementos se representan en el código con etiquetas, las cuales se encierran mediante los símbolos menor (**<**) y mayor (**>**) y, a su vez, pueden tener atributos definidos en forma específica.

En su estructura básica, vemos que un documento HTML tiene una primera línea de declaración de tipo de documento, un encabezado (**<head>**) y posteriormente un cuerpo (**<body>**).

En el encabezado, encontramos los elementos que describen el documento, como por ejemplo el título, y también etiquetas de metadatos (metadata) que pueden contener descripción, nombre del autor e idioma, entre otras opciones. También, en el encabezado, se suelen especificar los archivos que se incluyen o anexan al documento, como por ejemplo hojas de estilo o scripts que se ejecutan en la página del lado cliente.

SABER EL CORRECTO USO DEL HTML ES FUNDAMENTAL PARA TODOS LOS DESARROLLADORES.

En el cuerpo del documento, se definen los elementos que se representan en la pantalla del navegador. La mayoría de estos elementos poseen una apariencia predeterminada por defecto que, en algunos casos, puede tener alguna variación según el navegador. Si bien HTML permite definir algunas características de representación, como color y tamaño, no es recomendable utilizarlas ya que, para esta labor, contamos con las hojas de estilo, más conocidas como CSS.

La función de HTML es definir la estructura del documento, dejando de lado lo que se relaciona con representación. A la hora de diseñar y desarrollar nuestros proyectos, será muy importante pensar en como llevaremos adelante esa estrucutura antes de sentarnos a escribir las líneas de código que la conformarán.

El código HTML puede ser editado con cualquier editor de texto aunque, como veremos más adelante, hay herramientas que nos permiten trabajar con mayor comodidad y nos facilitan el trabajo.

En el **Capítulo 2** de este libro, analizaremos con mayor detalle los elementos que forman parte de HTML.

HTML MÁS ALLÁ DEL NAVEGADOR WEB

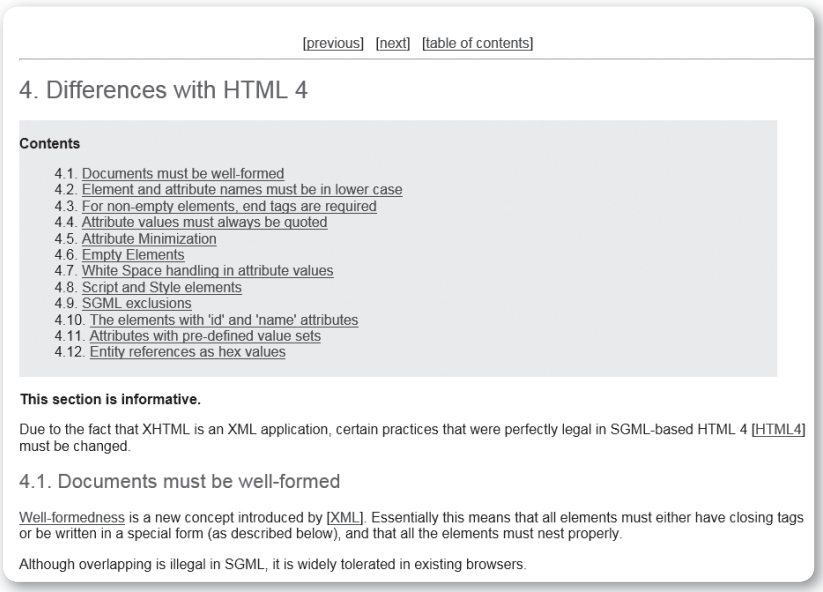
Si bien es habitual asociar a HTML solo con los navegadores web, es importante tener en cuenta que este lenguaje puede ser utilizado con cualquier software que sea capaz de interpretarlo. Un ejemplo de esto son los clientes de correo electrónico, que suelen soportar texto plano y también código HTML en los mensajes.

XHTML

Mientras HTML quedaba detenido en la versión 4.01, se fue desarrollando una alternativa que pudiera solucionar algunos de los problemas que se presentaban con este estándar. Así surgió XHTML, que es ni más ni menos que una adaptación de HTML al XML.
Si miramos el árbol genealógico, podremos observar, entonces, que de SGML surgen HTML y XML y que, de este último, nace XHTML, aunque en realidad, este toma prácticamente todas sus etiquetas de HTML.

La primera gran diferencia entre HTML y XHTML reside en que el primero es blando y permisivo, mientras que el segundo es estricto.

Para entender a qué nos referimos con esta rigurosidad de XHTML, veremos algunos ejemplos que nos aclararán el tema.


[previous] [next] [table of contents]

4. Differences with HTML 4

Contents

4.1. Documents must be well-formed
4.2. Element and attribute names must be in lower case
4.3. For non-empty elements, end tags are required
4.4. Attribute values must always be quoted
4.5. Attribute Minimization
4.6. Empty Elements
4.7. White Space handling in attribute values
4.8. Script and Style elements
4.9. SGML exclusions
4.10. The elements with 'id' and 'name' attributes
4.11. Attributes with pre-defined value sets
4.12. Entity references as hex values

This section is informative.

Due to the fact that XHTML is an XML application, certain practices that were perfectly legal in SGML-based HTML 4 [HTML4] must be changed.

4.1. Documents must be well-formed

Well-formedness is a new concept introduced by [XML]. Essentially this means that all elements must either have closing tags or be written in a special form (as described below), and that all the elements must nest properly.

Although overlapping is illegal in SGML, it is widely tolerated in existing browsers.

Figura 5. En **`www.w3.org/TR/xhtml1/diffs.html`**, podremos leer un documento con las diferencias entre XHTML 1.0 y HTML 4.

En XHTML, las etiquetas deben ser cerradas de acuerdo con el orden en que abren y según la forma en que están contenidos los elementos.

HTML permite hacer algo como lo que vemos en el siguiente código:

```
<p>Este es un texto con algunas <strong>palabras destacadas</p></strong>
```

En cambio, en XHTML es necesario cerrar primero el destacado ****, porque ese contenido está dentro del bloque de párrafo **<p>**. El ejemplo correcto en XHTML sería:

```
<p>Este es un texto con algunas <strong>palabras destacadas</strong></p>
```

Debemos tener en cuenta que otro de los aspectos que diferencia a HTML de XHTML reside en que, en este último, es imprescindible que los nombres de las etiquetas y de los atributos se escriban en minúscula, algo que resulta indistinto en HTML.

Veamos un ejemplo válido en HTML que no puede utilizarse en XHTML:

```
<H1>Esto es un titular</H1>
```

La forma correcta de escribir esto en XHTML es la siguiente:

```
<h1>Esto es un titular</h1>
```

En XHTML el valor de los atributos debe estar siempre entre comillas, mientras que esto no es indispensable en HTML.

El siguiente ejemplo es válido para HTML, pero no lo es para XHTML:

```
<p><a href=http://www.misitioweb.com>Mi enlace</a></p>
```

La alternativa correcta para XHTML es la siguiente:

```
<p><a href="http://www.misitioweb.com">Mi enlace</a></p>
```

DEL DIAL-UP A LA BANDA ANCHA WEB

El acceso a **Banda Ancha** ha tenido avances muy importantes a nivel mundial. Esto ha ocasionado que los usuarios con conexión del tipo **Dial-Up** sean cada vez menos. Esta evolución también impacta en lo que se refiere a las posibilidades de incorporar contenidos multimedia y al desarrollo de aplicaciones en Internet, que pueden ser utilizadas por un público cada vez mayor.

Otro aspecto importante en XHTML es que todas las etiquetas deben cerrarse. El caso típico es **
**, que sirve para indicar un salto de línea. En XHTML esta etiqueta debe cerrarse y se escribe **
** (un método abreviado para evitar tener que escribir **
</br>**).

En el siguiente capítulo de este libro, aprenderemos más sobre los elementos de HTML/XHTML y cómo indicarle al navegador el estándar utilizado. De esta forma lograremos dominar un poco más los conocimeintos que necesitamos para enfrentar un proyecto web.

HTML5

Al momento de decidir el camino por seguir en la evolución de HTML/XHTML, se planteó la controversia entre avanzar sobre XHTML 2.0 o bien realizar una nueva versión del lenguaje HTML. Finalmente, esta última idea fue la que triunfó y, por tal motivo, HTML5 es la versión que nos ocupa principalmente en este libro. Vale decir que HTML5 tuvo su primer borrador público a partir del año 2008.

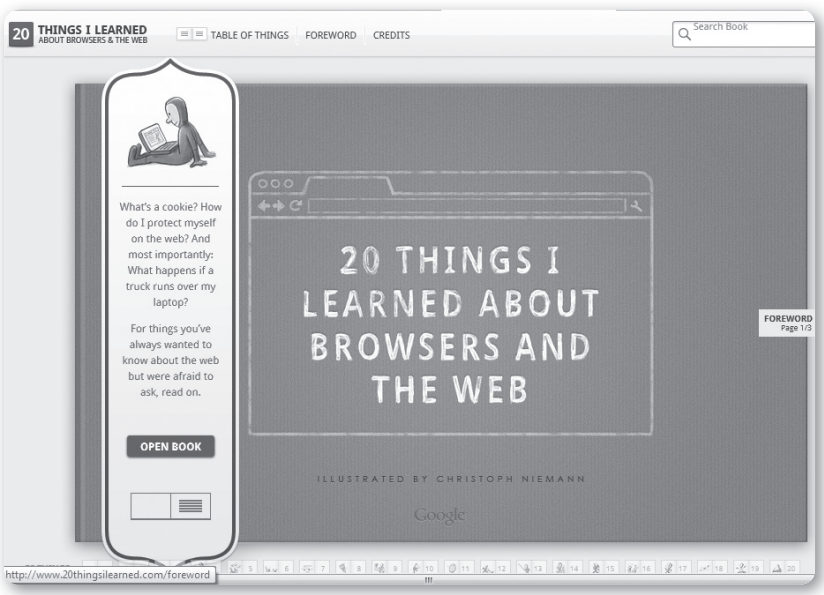


Figura 6. El potencial de HTML5 excede al desarrollo web y llega hasta los libros, como vemos en **www.20thingsilearned.com**.

HTML5 plantea una evolución necesaria para HTML, que luego de más de una década en la versión 4.01 necesitaba, de manera imperiosa, una renovación para estar al día con las necesidades del desarrollo web actual.

En HTML5, se destacan sus características semánticas, las posibilidades multimedia que incorpora, las nuevas funciones para formulario y las características que se definen para poder integrarse con tecnologías que permitirán abrir una nueva etapa en Internet, en lo que se refiere a la arquitectura de las aplicaciones. Por estos motivos, HTML5 es considerado como uno de los motores más importantes de la Web 3.0.

Es necesario destacar, para completar el camino de su evolución, que, en el mes de mayo de 2011, HTML5 ingresa en **Last Call**, y se ha anunciado que llegará a ser recomendación del W3C en el año 2014.

Otros lenguajes y tecnologías relacionados con el desarrollo web

Como veíamos en las páginas anteriores, conocer el lenguaje HTML es una de las bases para poder desarrollar la arquitectura de un sitio web. Pero su interacción con otros lenguajes y tecnologías es lo que realmente potencia el desarrollo y lo que nos permitirá lograr resultados acordes a las necesidades que requieren los proyectos web actuales.

Para poder aprovechar al máximo el potencial de HTML5 es fundamental también comprender el rol de las tecnologías que interactúan con este lenguaje de etiquetas y de qué manera deben integrarse.

A continuación, nos centraremos en conocer las características principales de CSS, JavaScript, AJAX y las tecnologías del lado servidor.

CSS

Las **hojas de estilo en cascada**, tal es su traducción del inglés *Cascading Style Sheets* (**CSS**), tienen como función establecer reglas de representación de un documento en un medio o dispositivo. Mediante estas reglas podremos establecer medidas, colores o cualquier otra

característica de representación de una página web, para que se vea reflejada en una pantalla de monitor, de un dispositivo móvil, una tablet, una impresora, un dispositivo braille o un televisor.

Figura 7. Crear una hoja de estilo distinta para la versión impresa de un sitio puede ayudar a mejorar la legibilidad del material.

La función principal de CSS es, por lo tanto, la de permitir separar el contenido y la estructura que se define en un documento HTML, de la representación, que queda a cargo de las hojas de estilos.

Esta separación es importante para un proyecto web ya que, además de permitir la definición de criterios que se deben respetar en el sitio, ofrece la posibilidad de que se definan clases para evitar la necesidad de rescribir código y, además, se pueden crear reglas para que el sitio se represente de una manera correcta en diferentes dispositivos.

Hay tres formas de incorporar los estilos a nuestro documento HTML.

La primera alternativa es hacerlo **inline**. Si bien es la más específica (más adelante en el libro veremos qué quiere decir esto) también es la menos recomendable para la mayoría de los casos. A continuación, se muestra el ejemplo del posicionamiento absoluto de un elemento con el estilo aplicado inline:

```
<div style="position: absolute; left: 20px; top: 50px;">
```

Otra opción consiste en definir los estilos directamente en el encabezado dentro de las etiquetas **<style type="text/css">** y **</style>**. Si bien es una alternativa más prolija que la anterior, no resulta muy útil porque la reusabilidad del código queda acotada a la propia página donde nos preocupamos de definir estos estilos.

La alternativa recomendable para el desarrollo web es la de trabajar con hojas de estilos externas (que se guardan en archivos con extensión **.CSS**) y luego incluirlas en el documento HTML mediante una línea de código en la cabecera, como vemos en el siguiente ejemplo:

```
<link rel="stylesheet" href="./css/style.css" type="text/css" media="screen" />
```

Nótese que, además de acceder a indicar la ruta en el **href**, también se procede a indicar el tipo de medio para el cual ha sido preparada la hoja de estilo; en este caso se trata de **screen**. Tengamos en cuenta que esto quiere decir que podemos contar con diferentes hojas de estilo para trabajar en el sitio y también para adaptar nuestro proyecto a la representación en distintas clases de dispositivos.

Si bien, en el **Capítulo 4**, profundizaremos sobre las características de CSS y cómo construir en este libro, a continuación veremos una breve historia de las hojas de estilo en cascada para conocerlas mejor y saber que versión de este estándar es conveniente usar.

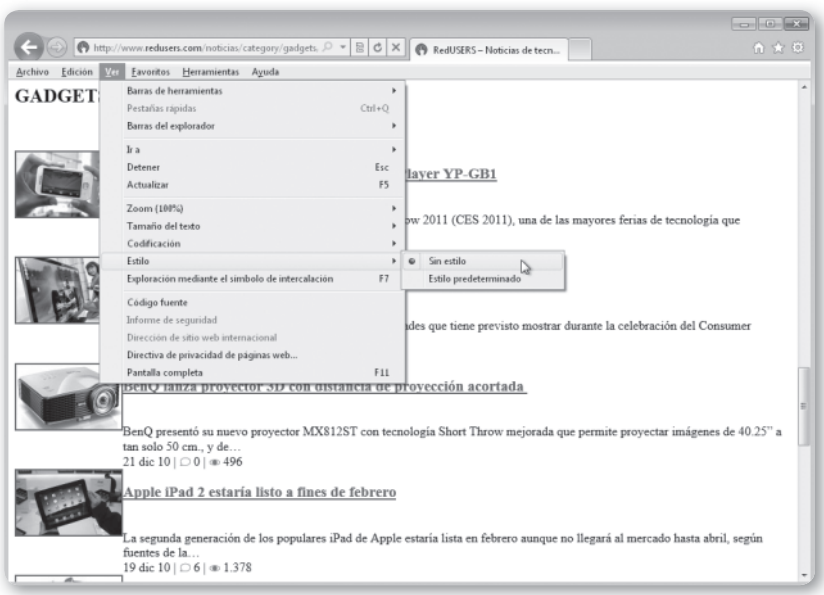


Figura 8. La mayoría de los navegadores ofrece la opción de desactivar CSS para poder visualizar el documento HTML libre de estilos.

Un poco de historia

El primer nivel de CSS se convierte en recomendación del W3C en el año 1996. Luego de bastante tiempo de trabajo, CSS2 se convierte en recomendación del W3C durante el año 1998.

Es importante tener en cuenta que se trata de una actualización que cuenta con muchas características relevantes, como las opciones de posicionamiento (fijo, relativo y absoluto), el uso del eje Z (z-index) y la posibilidad de definir distintos tipos de medio para las hojas de estilo (media types), entre otras capacidades adicionales.

Más adelante, llegaría la revisión 1 del nivel 2, conocida como CSS 2.1, que es la que se considera como estándar en la actualidad y es soportada por casi todos los navegadores modernos. Esta revisión incluye varias correcciones importantes, remueve algunas características obsoletas y agrega nuevas propiedades. Si deseamos conocer más sobre las especificaciones que brinda el W3C sobre CSS en su versión 2.1 podemos ingresar en el sitio que se encuentra en la dirección web **www.w3.org/TR/CSS21**.

Figura 9. Es importante tener en cuenta que una de las principales ventajas de utilizar CSS es la posibilidad de especificar distintos tipos de salidas para lograr compatibilidad con diversos dispositivos, por ejemplo, un teléfono móvil.

CSS3

Al tiempo que el nivel 2 de CSS (y sus respectivas revisiones) se desarrollaba, un equipo de trabajo ponía manos a la obra en lo que sería CSS3. Poco a poco se fueron conociendo sus principales características, que comenzaron a movilizar el mundo del diseño web. Sin ser un estándar aprobado aún, sus características más novedosas fueron pasando de un plano experimental entre los años 2009 y 2010, a ser utilizadas en muchos sitios web en la actualidad.

Figura 10. En Internet encontramos muchos sitios que han comenzado a utilizar CSS3 y HTML5, por ejemplo de **`www.thekillersmusic.com`**.

Entre las características más importantes que se destacan en este nuevo nivel de CSS encontramos las siguientes:

- Nuevas alternativas para dibujar bordes con el uso de opciones tales como color, imágenes, y radio o redondeado.
- Novedades en el trabajo con fondos, con el uso de degradados y la posibilidad de incluir múltiples imágenes.
- Uso de sombras para texto (text shadow).
- Se incluye la posibilidad de aplicar sombra a elementos (box shadow).

- Novedades en cuanto al uso del color y de la opacidad.
- Incorporación de muchas novedades en lo que se refiere a flujo de texto dentro del sitio (text overflow).
- Nuevas características para trabajo con múltiples columnas.
- Finalmente se le dice adiós al problema de las limitaciones con las tipografías, con el uso de @fontface.
- Novedades y algunos cambios en el uso de pseudoelementos.
- Características relacionadas con la interfaz de usuario.
- Capacidad de rotación de elementos.
- Opciones de transformación de elementos.
- Incorporación de transición y también funciones de animación.

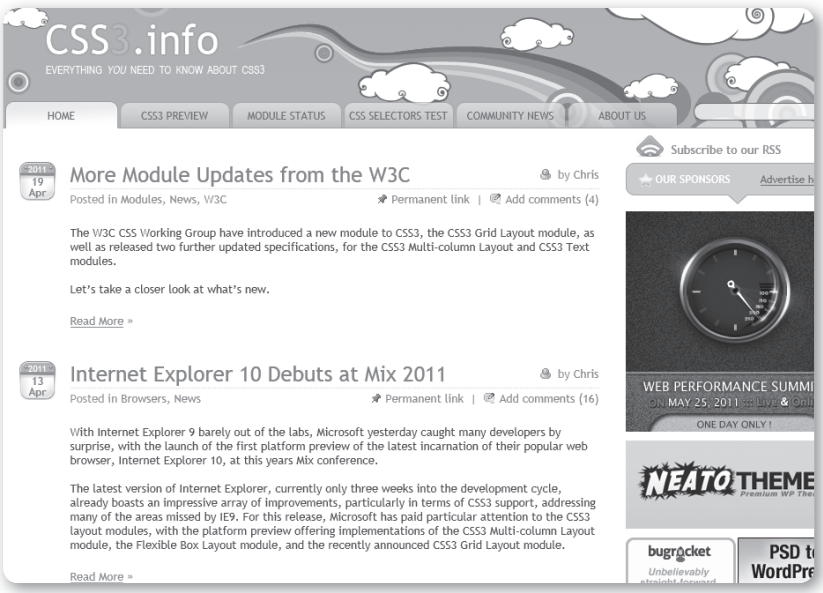


Figura 11. Si ingresamos en **`www.css3.info`**, encontraremos un sitio con muchas novedades sobre CSS3.

En el **Capítulo 4** de este libro, ahondaremos sobre cada una de las características que se incorporan a partir del nivel 3 de CSS y veremos cómo influyen en la forma de diseñar y desarrollar un sitio web. Está claro que CSS3 plantea una revolución en las técnicas conocidas y abre un muy interesante abanico de posibilidades para la Web.

JavaScript

Es importante tener en cuenta que **JavaScript** (dialecto de ECMAScript) es un lenguaje multiparadigma que requiere de un intérprete para ser ejecutado. Así como los navegadores web cuentan con un motor para representar el contenido de HTML y CSS, también tienen un motor que funciona como intérprete para el código JavaScript.

Como lenguaje, JavaScript salió a la luz en el año 1995. Luego de pasar por algunos nombres y denominaciones, se fue convirtiendo en una alternativa para programación del lado cliente.

Su finalidad principal es permitir la creación de páginas dinámicas, con código que puede ejecutarse desde el lado cliente, alivianando la tarea del servidor y disminuyendo la cantidad de peticiones que se le hagan. Por sus características, resulta útil para validación de formularios, mostrar y aplicar efectos, y exhibir avisos en pantalla.

Es importante remarcar que su uso cobró mayor fuerza a partir del éxito de AJAX y el importante impulso de la Web 2.0.

La inclusión de un código JavaScript dentro de un documento HTML puede realizarse ubicándolo en el encabezado entre las etiquetas **<script type="text/javascript">** y **</script>**.

Si bien esta alternativa sigue siendo utilizada, la mejor opción es externalizar el contenido de JavaScript en un archivo de extensión **.JS**. Luego lo incluimos en el documento HTML con una línea de código en el encabezado como vemos en el siguiente ejemplo:

```
<script type='text/javascript' src='./js/script.js'></script>
```

En el valor de la propiedad **src**, se debe especificar la ruta y el nombre del archivo que contiene el código JavaScript que se desea incluir.

Algunas consideraciones y características para tener en cuenta

JavaScript es un lenguaje que puede escribirse directamente en cualquier editor de texto; será suficiente solo con guardar el archivo con extensión **.JS** ya que no requiere de un compilador. Claro está que, como veremos más adelante, existen ciertas aplicaciones que nos facilitan el trabajo con ayuda de la sintaxis y otras características visuales.

Las tabulaciones, los retornos de carro (es decir los **ENTER**) y los

espacios en blanco agregados de más son ignorados por JavaScript, aunque es recomendable utilizar esto con criterio para una mejor legibilidad del código. También es importante saber que JavaScript es case-sensitive, o sea, tiene en cuenta las mayúsculas y las minúsculas.

Otro aspecto para tener presente consiste en que JavaScript no requiere que las sentencias concluyan con **;** (punto y coma). Sin embargo, hacerlo de esta forma ayuda a la legibilidad.

Las variables que se utilizan en JavaScript no tienen distinción de tipo; por tal motivo, en una misma variable es posible ir asignando diferentes tipos de valores, sin necesidad de indicaciones adicionales. Veamos un ejemplo de declaración e inicialización de una variable:

```
var valor_1 = 100;
```

También es posible declarar primero la variable y, luego, asignar un valor en otra parte del código, si así lo requiere la estructura que estamos programando. Además, es posible obviar el paso de la declaración de la variable, aunque según el tipo de aplicación que se trate de construir, esto puede ser poco recomendable.

Las variables pueden ser locales o globales, y esto dependerá del ámbito en el cual sean declaradas (dentro o fuera de una función).

En el caso de los arrays, en JavaScript se los puede ver como un conjunto o colección de variables. Veamos un ejemplo:

```
var nombres = ["Julio", "Marcelo", "Mario", "Pablo", "Claudio", "Diego",
"Horacio", "Manuel"];
```

Debemos tener en cuenta que los arrays comienzan desde la posición cero. Si deseamos asignar a una variable el primer valor del ejemplo anterior, debemos tener en cuenta que es **0**.

JAVA Y LA DIFERENCIA CON JAVASCRIPT

Aunque algunos pueden confundirlos, **Java** y **JavaScript** no son lo mismo. Java es un lenguaje que se basa en el paradigma de programación orientada a objetos y que requiere ser compilado como applet para ser cargado en una página web. Es una solución multiplataforma, pero, para poder ejecutar esta mini aplicación, el equipo cliente debe tener cargada la **Máquina Virtual de Java**.

```
var primernombre = nombres[0];
```

Al realizar tareas de programación más avanzadas con JavaScript, se hace imprescindible el uso de funciones, las cuales nos permiten definir tareas que luego utilizaremos e, incluso, podremos reutilizar. Las funciones de JavaScript se definen con la palabra reservada **function**, luego se especifica el nombre de la función y, entre paréntesis, los argumentos que puede recibir (si no recibe ninguno, el paréntesis se escribe vacío). A continuación, entre corchetes se escribe lo que debe realizar la función. En el siguiente ejemplo veremos cómo se puede efectuar una sencilla función de suma:

```
 function  (numero1, numero2) {
var resultado = numero1 + numero2;
        return resultado;
 }
```

Para llamar la función, debemos hacerlo en el lugar del código donde deseemos que se ejecute (siempre dentro de las etiquetas script de JavaScript). Siguiendo con el ejemplo anterior, llamaremos a la función creada, le pasaremos dos números para que los sume y mostraremos el resultado en la pantalla con una ventana de alerta.

```
alert(suma(10, 5));
```

Debemos tener en cuenta que con JavaScript, se pueden capturar eventos que se produzcan en el documento HTML para realizar alguna función específica que deseemos establecer. En los próximos capítulos, hablaremos sobre los eventos y sus características.

AJAX

Al hablar de **AJAX**, debemos comenzar por comprender que no es un lenguaje de programación, como JavaScript o PHP, sino que representa una técnica que reúne a un conjunto de tecnologías y lenguajes para crear lo que se conoce como **RIA** (*Rich Internet Applications*). Su nombre se comenzó a utilizar en 2005, en pleno auge de la Web 2.0; sin embargo, los

lenguajes y las tecnologías que son utilizados con AJAX ya existían en ese momento. Lo que ocurrió es que, por las necesidades de desarrollo que comenzaron a florecer en aquella época, se hizo necesario un cambio en las técnicas empleadas, y allí fue donde AJAX encontró su lugar.

El término AJAX es un acrónimo que proviene de *Asynchronous JavaScript And XML*, que, al castellano, podría traducirse como JavaScript asíncrono y XML. Justamente este es el punto fuerte de AJAX: poder trabajar con datos de manera asincrónica, valiéndose de JavaScript como lenguaje del lado cliente para manejar datos que le llegan desde el servidor. De esta manera, el motor de AJAX trabaja como un intermediario entre el cliente y el servidor, pero, en lugar de demorar procesos, los administra de tal manera que es posible, por ejemplo, la recarga de solo algunas partes de una página web. Esta posibilidad cambia el paradigma de la necesidad de una recarga completa de la página y permite construir aplicaciones web más potentes, emulando incluso a muchas de las soluciones que se veían posibles solo en software de escritorio.

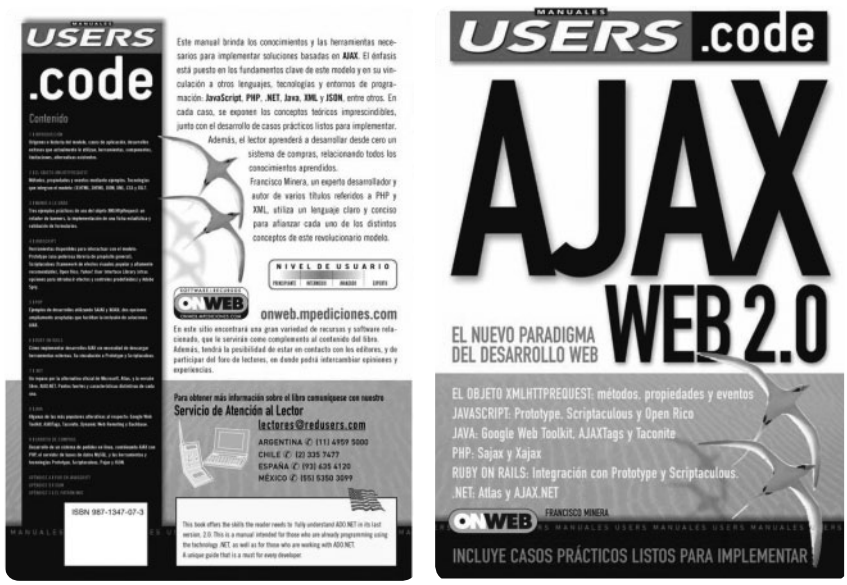


Figura 12. Para conocer más sobre esta tecnología, es recomendable leer AJAX Web 2.0, del autor Francisco José Minera.

Los lenguajes y tecnologías que intervienen en AJAX son:

- **HTML/XHTML y CSS**: ya hemos hablado bastante de ellos, son los que permiten la representación en el navegador.
- **XML y JSON**: son los que permiten realizar el intercambio de datos y también efectuar la manipulación de estos.
- **XMLHttpRequest**: es el que permite realizar el intercambio asíncrono de los datos que sean necesarios.
- **DOM**: es la interfaz que permite acceder a las partes de un documento como si fueran objetos y, a partir de eso, modificarlas.
- **JavaScript**: es el lenguaje utilizado para hilvanar todo.

XMLHttpRequest

Como su nombre lo indica, **XMLHttpRequest** es una interfaz que permite realizar peticiones a servidores web mediante el protocolo HTTP (y también utilizando el protocolo HTTPS).

Además de XML, el formato de transferencia puede ser codificado en texto plano, HTML y JSON. XMLHttpRequest es uno de los elementos fundamentales para permitir la transferencia asincrónica de datos de AJAX.

Esta especificación fue creada por Microsoft y se incorporó en Internet Explorer a partir de la versión 5 (usando ActiveX).

Posteriormente fue incorporada por los navegadores Mozilla, Safari y Opera, entre otros. Al ser un navegador más joven, Google Chrome lo utiliza desde el momento de su nacimiento.

Para conocer más sobre esta especificación, podemos ingresar en el sitio web que se encuentra en **www.w3.org/TR/XMLHttpRequest**.

XMLHttpRequest Level 2 se encuentra actualmente en Draft la vemos en: **http://www.w3.org/TR/XMLHttpRequest2**.

APLICACIONES: WEB VS. ESCRITORIO

Los avances de las tecnologías web posibilitan crear aplicaciones cada vez más complejas. Hace una década, parecía muy complicado desarrollar aplicaciones que pudieran correr en el navegador y que tuvieran las características y la potencia de las de escritorio. Hoy en día, ese límite se está borrando, y cada vez son más robustas las aplicaciones web que se encuentran en el mercado.

DOM

Bajo el nombre de **DOM** (*Document Object Model*) se define una interfaz que permite la representación del documento en el modelo de objetos. De esta manera, es posible acceder, por ejemplo, a los elementos contenidos en un documento HTML o XML con la posibilidad de manipularlos (crearlos, borrarlos o modificarlos).

En palabras sencillas, DOM nos permite tener acceso a un documento (que puede ser una página web) representado en una estructura de objetos. Es posible modificar estos objetos, cambiarles sus atributos o bien agregarles nuevos. El uso de DOM es una de las características principales que le da mayor funcionalidad a AJAX.

Utilizando JavaScript, es posible acceder de diferentes formas directamente al elemento, como por ejemplo, mediante su id con **`getElementById()`** o a través de su etiqueta con **`getElementsByTagName()`**. Vale decir que al acceder a los elementos del documento mediante DOM, podremos manipular su contenido y también sus atributos. Así podremos percibir cambios en la página sin recargarla por completo.

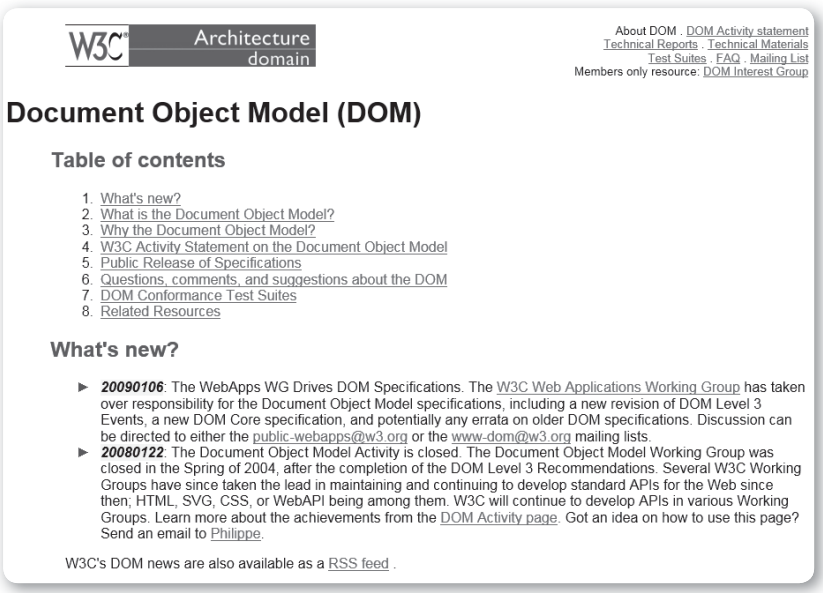


Figura 13. Podremos encontrar más información sobre Document Object Model en la dirección web **`www.w3.org/DOM`**.

JSON

JavaScript Object Notation (**JSON**) es la denominación en inglés que recibe el formato de notación literal de objetos, el cual es utilizado principalmente para el intercambio de datos en AJAX.

En la actualidad, se emplea mucho en aquellos proyectos donde, con frecuencia, se produce un gran nivel de flujo de datos asíncronos entre cliente/servidor, ya que permite tener la información organizada y con facilidad de acceso para ser manipulada.

Entre estas ventajas se encuentra la simplicidad con la que permite interactuar con arrays y el uso de objetos. Por sus características, en especial su simpleza y eficiencia, JSON se ha convertido en una muy interesante alternativa a XML, como opción de codificación de datos en AJAX. Pero también resulta importante destacar que, en un mismo desarrollo, es posible utilizar tanto la alternativa para el manejo con XML con XMLHttpRequest, como la solución que nos ofrece JSON. Es importante tener en cuenta que la mayoría de los navegadores modernos cuenta con soporte nativo para JSON.

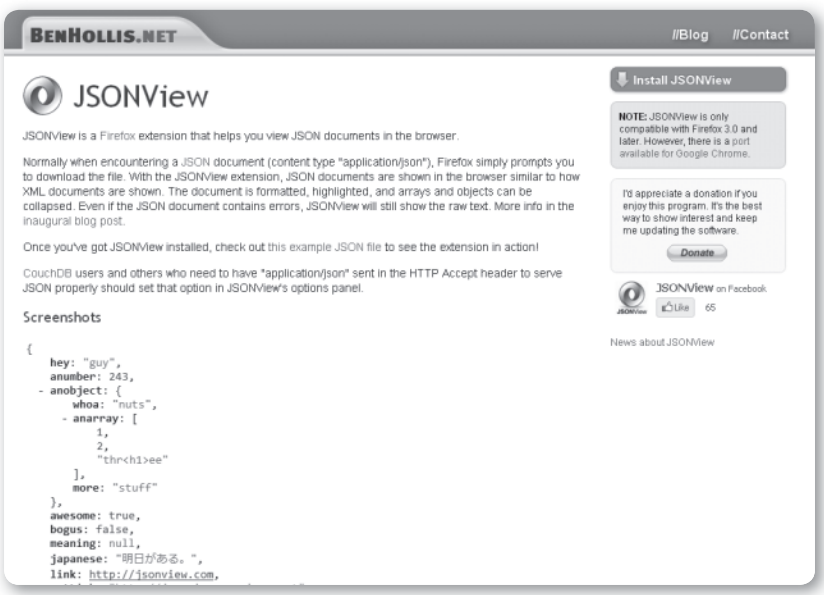


Figura 14. JSONVlew (**`http://jsonview.com`**) es una extensión para Firefox que permite visualizar documentos JSON en el navegador.

Entre los lenguajes que pueden trabajar con JSON, encontramos JavaScript, Java, PHP, Perl y Ruby, entre otros.

Si estamos interesados en saber más sobre las opciones y la forma de trabajar con JSON, podemos ingresar en el sitio web que se encuentra en la dirección **www.json.org/json-es.html**.

RIA

Rich Internet Applications (**RIA**) es el término con que se denomina a las aplicaciones de Internet enriquecidas. En pocas palabras, este tipo de aplicaciones tienen características similares a las de escritorio, pero corren directamente en el navegador.

Aunque existen alternativas, debemos tener en cuenta que la ventaja de desarrollar RIA con AJAX es que el usuario no necesita instalar software adicional en su equipo. Tan solo necesita contar con un navegador moderno y una buena conexión a Internet; de esta forma estará listo para correr la aplicación web.

Figura 15. Las aplicaciones RIA, además de construirse con AJAX, también pueden ser realizadas con Adobe Flash y Adobe Flex.

Este tipo de aplicaciones, surgidas a partir de la Web 2.0, son cada vez más potentes y, por lo tanto, ofrecen soluciones muy eficaces para hacer frente a distintos tipos de necesidades.

Frameworks

El concepto de **framework** es muy usual en el ámbito de programación y ha cobrado gran importancia en lo que se refiere a desarrollo web.

Debemos saber que, en líneas generales, un framework es un fragmento de código que cuenta con soluciones para enfrentar una necesidad de desarrollo en particular, resolviendo cuestiones de bajo nivel para simplificar la labor del programador que lo utiliza.

En AJAX, encontramos varias librerías muy interesantes que simplifican nuestro trabajo con esta tecnología.

Entre las más importantes, podemos mencionar a jQuery (que se encuentra en la dirección **http://jquery.com**), Prototype (**www.prototypejs.org**) y MooTools (**http://mootools.net**), entre otras.

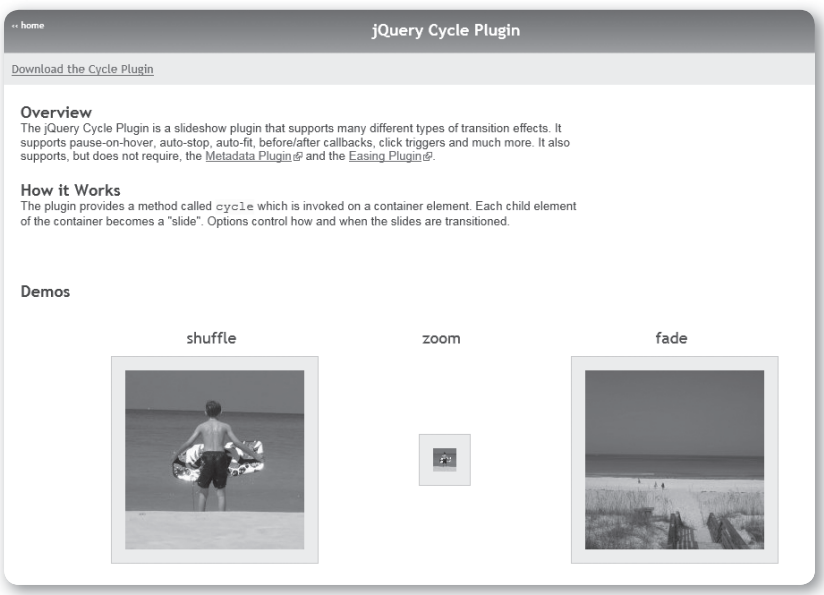


Figura 16 . Varios frameworks cuentan con plugins que ofrecen funcionalidades personalizadas, como **`jQuery Cycle Plugin`**.

Tecnologías y lenguajes del lado servidor

Es importante saber que los lenguajes y tecnologías que se encargan de trabajar del lado servidor requieren de un intérprete que devolverá al cliente (el equipo que utiliza el usuario para acceder a los recursos) el resultado que pueda ser comprendido por el navegador.

PHP es uno de los lenguajes del lado servidor más utilizados en la actualidad y, combinado con técnicas AJAX, nos permite crear aplicaciones web de diferentes grados de complejidad.

PHP se destaca por ser un lenguaje libre y multiplataforma, lo cual resulta una gran ventaja, ya que puede funcionar tanto en servidores Windows como Linux que cuenten con el software apropiado.

Aunque se lo considera multiparadigma, en sus últimas versiones ha reforzado el concepto de orientación a objetos.

Navegadores

Ya hemos explicado la importancia de los navegadores para el mundo web: se trata de las aplicaciones que reciben la información, la interpretan y finalmente la muestran al usuario.

Existen navegadores para diferentes dispositivos y plataformas. Cada uno con sus propias características, todos tiene la misma finalidad: ofrecerle al navegante una representación lo más fiel posible del recurso al que está accediendo, que, por lo general, será un sitio o una aplicación web. Existen diferentes navegadores, pero nos centraremos en los principales: Internet Explorer, Firefox, Chrome, Apple Safari y Opera.

ACTIVE SERVER PAGES (ASP)

ASP (*Active Server Pages*) es una tecnología que permite el desarrollo de páginas web que se procesan e interpretan del lado servidor. Una vez realizada la petición, el intérprete devuelve un resultado al cliente, quien podrá visualizarlo en el navegador. ASP es una tecnología desarrollada por Microsoft, que desde hace un tiempo ha comenzado a ser reemplazada por ASP.NET.

Internet Explorer

La primera versión de **Internet Explorer** (IE) apareció en el año 1995. Eran tiempos de Windows 95, y lo que hoy conocemos como Internet aún estaba en pañales. La versión 2 salió a la luz hacia fines de 1995, y la 3 se publicó en 1996, con mejoras en lo que se refiere a la compatibilidad con VBScript, JavaScript y también CSS.

Internet Explorar 4 se lanza en el segundo semestre de 1997, mejorando la compatibilidad con HTML y CSS. Ya eran tiempos de Windows 98, aunque esta versión aún contemplaba compatibilidad hacia atrás con los sistemas operativos de Microsoft.

En 1999, con la llegada de Windows 98 SE, aparece IE 5, que ofrece compatibilidad con CSS2. Con Windows XP, llega IE 6. Corría el año 2001 y, entre ese año y los siguientes, Microsoft logró una cuota del mercado de los navegadores, hasta superar picos de 90% de usuarios.

Internet Explorer 6 corregía errores y mejoraba la compatibilidad respecto de las recomendaciones del W3C. Aunque hoy, pasados los años, resulta un navegador que se considera obsoleto, en su momento ofreció varias mejoras para la familia de browsers de Microsoft.

Después de varias betas y un importante tiempo de desarrollo, en 2006, con el lanzamiento de Windows Vista, llega Internet Explorer 7.

Más seguro y estable, ofrece varias mejoras respecto de la versión anterior, en especial, en lo que se refiere a compatibilidad con CSS, navegación por pestañas, soporte a transparencias en archivos PNG, correcciones de bugs y renderizado de las páginas.

Aunque para lo que hoy en día se espera en Internet puede resultar anticuado, y, por lo tanto, aún existe una porción considerable de usuarios que no han migrado y por lo tanto lo siguen utilizando.

Internet Explorer 8 se lanza en el primer semestre del año 2009 y es el último de la familia de navegadores de Microsoft en prestar

RESOLUCIÓN DE PANTALLA DEL USUARIO

Tengamos en cuenta que la gran variedad de dispositivos y también de opciones de pantalla para los equipos de escritorio ha hecho que, hoy en día, sea muy amplio el espectro de posibilidades, en lo que se refiere a la resolución que puede tener el usuario. Por esta razón, es necesario verificar la correcta visualización de nuestro sitio en diferentes resoluciones de pantalla.

compatibilidad con el sistema operativo Windows XP. Por su parte, **Internet Explorer 9** se lanza en el año 2011 y, a partir de esta versión, llega la compatibilidad con CSS3 y HTML5. IE9 también mejora las funciones de soporte para DOM, SVG y JavaScript.

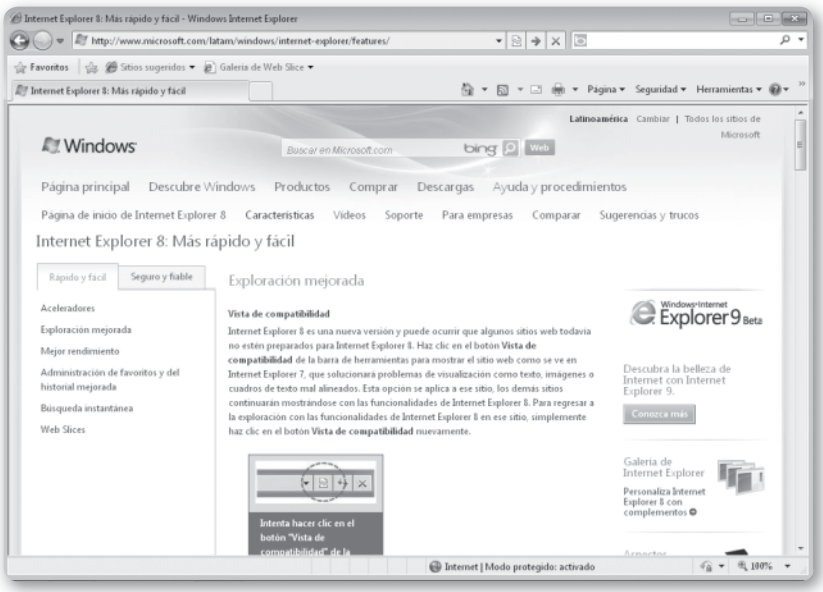


Figura 17. Internet Explorer 8 se destaca por mejorar la compatibilidad con CSS 2.1, pero no ofrece soporte para CSS3 y HTML5.

Cabe decir que, aunque ya no tenga el dominio absoluto que supo tener en el mercado de los navegadores, aún hoy toda la familia de navegadores Internet Explorer, sumados, resultan los más utilizados por los usuarios. Por este motivo, es muy importante realizar pruebas en las últimas versiones para asegurar la compatibilidad de nuestros proyectos.

La **Platform Preview** de **Internet Explorer 10** fue lanzada también en 2011, y se esperan mejoras en esta versión por el lado del soporte a HTML5 y CSS3, especialmente en lo que se refiere a multicolumna y al módulo denominado Flexible Box Layout.

Si deseamos obtener la última versión final de Internet Explorer en español, se puede acceder a la siguiente dirección web: **www.microsoft.com/spain/windows/internet-explorer**.

Mozilla Firefox

La primera versión de **Firefox** salió a la luz en noviembre de 2004. Un año después y con varios cambios, llegaría la versión 1.5.

La navegación mediante pestañas y el gestor de complementos serían algunas de las mejoras de la versión 2, que fue publicada en 2006.

En 2008, con la versión 3, llegan varias correcciones y ajustes en lo que se refiere al cumplimiento de estándares web. Un aluvión de descargas en las primeras horas acompañaría el éxito de esta versión.

Figura 18. Mejoras importantes en el rendimiento hacen de Firefox 4 un navegador preparado para las exigencias de la Web actual.

A partir de la versión 3.5, publicada en el año 2009, se comienza a dar soporte a algunas características de HTML5, como las etiquetas de **<audio>** y **<video>** (con compatibilidad nativa con los códecs **Ogg Theora** y **Ogg Vorbis**). En esta versión, también hay mejoras por el lado de JavaScript con el motor TraceMonkey.

En 2010, llega la versión 3.6, que durante ese mismo año recibiría varias actualizaciones hasta llegar a la 3.6.13. Entre las mejoras más importantes que se pueden mencionar, están las relacionadas con

JavaScript, el soporte a Web *Open Font Format* (WOFF), soporte a nuevos atributos de CSS3, DOM y HTML5 (File API y Drag & Drop API).

También en 2010, aparece la beta de Firefox 4, cuya versión definitiva se publica en marzo de 2011. Con Firefox 4, llegan mejoras en lo que se refiere a audio y video; con el soporte a video HD con WebM, se admite también el formato WebGL para gráficos 3D en la Web, mejoras en el soporte a CSS3, soporte a elementos y atributos de formulario de HTML5, soporte multi-touch para Windows 7, SVG como archivos de imágenes y fondos.

En la versión 5 de Firefox, lanzada en junio de 2011, se actualiza la interfaz, el trabajo con pestañas y su menú contextual con más funciones integradas; también hay mejoras en la gestión de cuentas de usuarios. Su lanzamiento fue anunciado para mediados de 2011.

FIREFOX SE DESTACA POR LA CANTIDAD DE ADD-ONS ÚTILES PARA EXTENDER SUS FUNCIONES.

Para descargar la última versión final disponible de Mozilla Firefox, podemos ingresar en **www.mozilla-europe.org/es/firefox**.

Google Chrome

La primera versión de **Google Chrome** fue lanzada por Google en el año 2008 y estaba pensada para plataforma Windows (XP y superiores).

La versión 2 de este navegador llegó en el primer semestre del año 2009 para ofrecer una mayor cantidad de idiomas.

Con la versión 3, publicada en septiembre de 2009, llega el soporte a las etiquetas **<canvas>**, **<audio>** y **<video>** de HTML5. También se mejora la performance en el motor correspondiente a JavaScript.

Con la versión 4, publicada en enero de 2010, aparecen también

UNIFORM RESOURCE LOCATOR (URL)

Al manejarnos en el mundo del desarrollo Web, es importante saber que cuando se hace referencia a una **URL** (*Uniform Resource Locator*) se está hablando de una cadena de caracteres que hacen posible identificar un determinado recurso informático (habitualmente alojado en un servidor). En el uso cotidiano, este término suele emplearse para hacer referencia a la dirección donde se encuentra un archivo en Internet.

las betas para otros sistemas operativos (Linux y Mac). En lo que se refiere a la versión Windows, se mejora el rendimiento general, se suma soporte para una gran cantidad de extensiones y, en lo vinculado a HTML5, se destaca el soporte para almacenamiento local.

Google Chrome 5 se publica en el mes de mayo de 2010. Este lanzamiento ya ofrece las versiones definitivas tanto para Windows como para Mac y Linux. Drag&Drop, geolocalización, App Cache y WebSockets son algunas de las características relacionadas con HTML5 que ya se pueden comenzar a utilizar con Google Chrome 5.

Figura 19. La última versión de Google Chrome se puede descargar desde la dirección **`www.google.com/chrome?hl=es`**.

En el mes de septiembre de 2010, se publica la versión 6, que cuenta con varias mejoras en las características de sincronización del navegador. También se destaca por tener el plugin de Flash Player 10.1 incorporado y activado de manera predeterminada. En lo vinculado con HTML5, se da soporte a WebM para video.

A partir de la versión 7, se comienza a notar con mayor fluidez el ritmo de actualización del navegador de Google, que llega a lanzar una

versión cada seis semanas. En lo referente a HTML5, Google Chrome 7 incorpora varias mejoras, entre las que se destacan el soporte a la File API y una performance notable en los tests de HTML5.

Google Chrome 8 se lanza en diciembre de 2010 y principalmente brinda correcciones de bugs, mejoras de performance y el lector de PDF incorporado activado por defecto.

La versión 9 del navegador de Google nos trae WebGL activado de manera predeterminada y aislamiento del proceso de Flash Player, entre otras características de interés.

En la versión 10, se destacan las mejoras en el rendimiento, y es más veloz que su antecesor gracias a las actualizaciones realizadas a los motores de renderizado y de JavaScript. En las siguientes versiones, se mantiene la apuesta por mejorar el rendimiento, aprovechar al máximo las características de hardware, corregir errores y brindar mayor compatibilidad a las nuevas características de HTML5 y también de CSS3.

> CHROME 12, LANZADO EN 2011, INCORPORA ACELERACIÓN POR HARDWARE PARA 3D CSS.

La versión 11 incluye compatibilidad con la API de voz (*Speech to text* API) para permitir que el usuario ingrese un texto mediante voz.

En su corta historia, Google Chrome ha evolucionado mucho en su arquitectura y tecnología, pero no solo eso, sino que también se ha convertido en el navegador preferido por muchos usuarios, aunque aún se encuentra por detrás de IE y de Firefox en las preferencias del público. Chrome se destaca por su frecuencia de actualización.

Safari

Desarrollado por **Apple**, este navegador que primero nació para sistemas Mac, luego supo pegar el gran salto para plataformas Windows y también cuenta con versiones para móviles (iPhone), reproductores multimedia (iPod Touch) y tablets (iPad).

La primera versión apareció como beta a principios del año 2003 y, a mediados de ese año, fue lanzada en forma definitiva. En octubre de 2003, con la llegada de la versión 1.1, Safari comenzó a ser el navegador por defecto de los sistemas Mac OSX, lo cual significó un impulso muy importante para este software.

La versión 2 llegaría en el año 2005, y la versión 3 sería lanzada dos años después, ya con el soporte tanto para Mac OSX como para Windows.

Safari 4 se publica en 2009 con la gran novedad en su motor de JavaScript (Nitro). En 2010, llega Safari 5 con muchas novedades en lo que se refiere al soporte de HTML5: pantalla completa y subtítulos para **<video>**, geolocalización, EventSource, WebSocket, Drag&Drop, etiquetas semánticas y atributos de formularios, entre otras características.

Figura 20. En navegador Safari, en la actualidad, es el browser por defecto de los sistemas Mac OSX, iPhone, iPod Touch e iPad.

Opera

Si hay algo que destaca a **Opera**, es que podemos encontrar versiones para diferentes sistemas y dispositivos, entre ellos, equipos de escritorio, smartphones, PDAs y consolas de videojuegos. Aunque Opera no se encuentra en el podio entre los navegadores más utilizados de la actualidad, cuenta con varias características muy interesantes, que a lo largo de su historia han demostrado que es un navegador de vanguardia.

La primera versión de Opera salió a la luz en 1995, pero, curiosamente,

no se lanzó para el público general. De esta manera, en el año 1996, la versión 2 es la primera en estar disponible para todos los usuarios.

En 1997, se publica la versión 3 y, un año después, se lanza la 3.5, con una destacada compatibilidad con CSS.

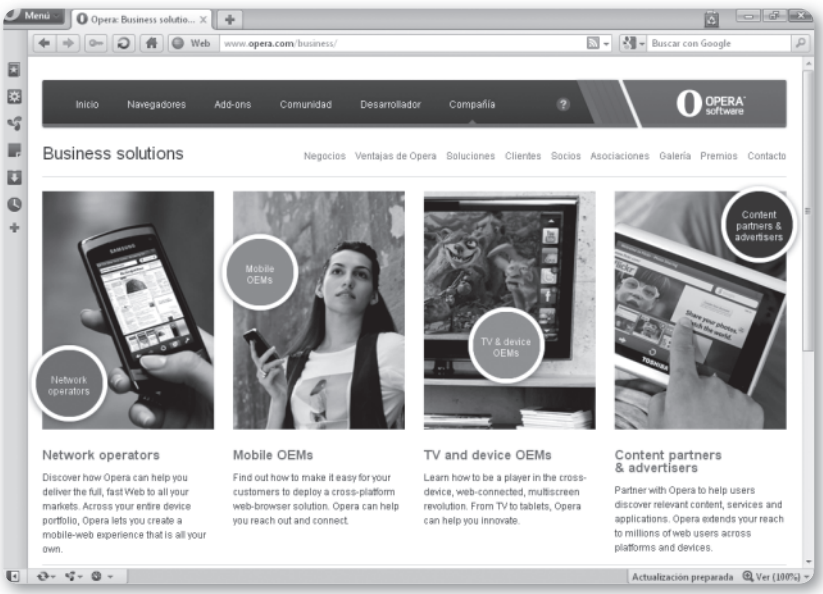


Figura 21. Una de las características que más sorprende en Opera es la variedad de dispositivos y sistemas en los que es soportado.

La versión 4 llega a mediados del año 2000; esta se destaca por su buen soporte y compatibilidad con CSS1, CSS2 y HTML4. En las versiones para móviles, se incluye soporte para WAP y WML. Con Opera 4, también llegan las primeras versiones para Linux y Mac.

Opera 5 es lanzado a fines de 2000 e incluye un cambio en la licencia; esta versión muestra un espacio con publicidad (que se puede remover si se opta por la versión paga). La versión 6 se presenta en noviembre de 2001 e incorpora importantes cambios en la interfaz.

Opera 7, lanzado en el año 2003, es el primer navegador de esta familia que incluye el motor de renderizado **Presto**. Con este cambio, también llegan mejoras en la compatibilidad con los principales estándares web. Esta versión ofrece uno de los saltos tecnológicos más

importantes de este navegador. Opera 8, de 2005, se destaca por su interfaz gráfica simplificada y las mejoras en lo que se refiere a la compatibilidad con JavaScript y AJAX.

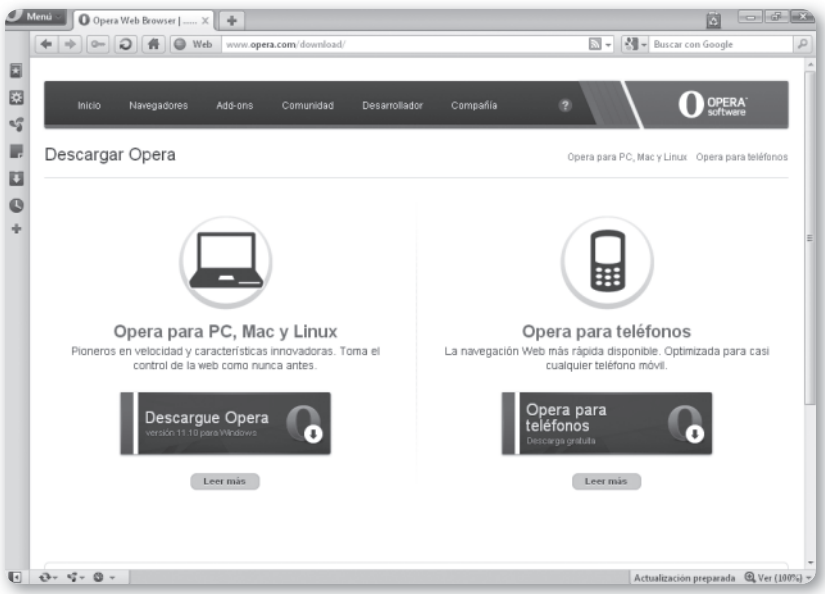


Figura 22. Para acceder a las opciones de descarga de Opera, podemos ingresar en la dirección **`www.opera.com/download`**.

Con el navegador web Opera 8.5, lanzado en septiembre de 2005, llega un nuevo cambio en lo que se refiere a licencia, ya que es la primera versión en pasar a modalidad **freeware**.

La versión 9 llega con mejoras en cuánto a soporte SVG 1.1 Basic, aspectos relacionados con la seguridad y el manejo de widgets.

La versión 10 de Opera fue lanzada en 2009 con cambios en la interfaz y también en la compatibilidad con los estándares web. Durante el tiempo en que recibió sucesivas actualizaciones, fue mejorando el soporte a HTML5; en especial se destaca en lo referido a atributos de formulario, aspecto en el que logra resultados muy notables.

Opera 11, lanzado a fines del año 2010, vuelve a poner manos sobre la simplificación de acceso para el usuario, las posibilidades de personalización, la seguridad y la performance del producto; de esta

forma se presenta como una alternativa más que eficiente. En esta versión, también se incluyen mejoras al soporte de HTML5 y CSS3, destacándose lo relacionado con WebSockets y geolocalización.

Los motores de renderizado

Los motores de renderizado son el verdadero corazón del navegador. De ellos depende la representación final de los elementos de una página web, y, en ellos, también recae la palabra **compatibilidad**.

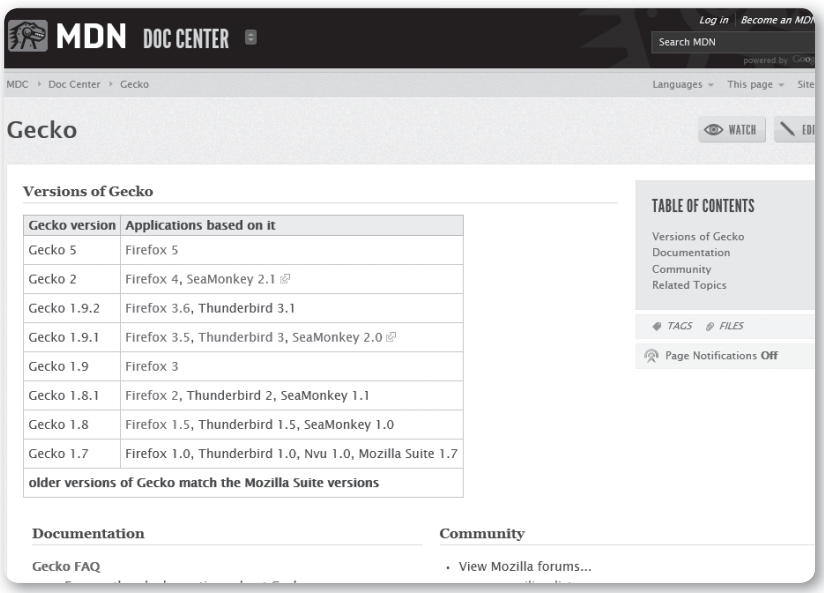


Figura 23. Podemos encontrar información sobre Gecko (motor de Firefox) en **`https://developer.mozilla.org/en/Gecko`**.

En pocas palabras, el motor de renderizado es un componente interno del navegador que tiene la capacidad de tomar el contenido HTML y CSS y representarlo en el dispositivo de salida (por lo general, la pantalla).

Trident es el nombre del motor de renderizado que se incluye en Internet Explorer desde la versión 4 del navegador de Microsoft.

Debemos tener en cuenta que la versión 5.0 de Trident llega junto a Internet Explorer 9, y, en ella, podemos observar que se destaca el tan esperado soporte para HTML5 y también para CSS3.

Gecko es el motor que acompaña a Firefox y también a otros navegadores no tan difundidos (varios de entorno Linux). Es mantenido por la Fundación Mozilla y la Corporación Mozilla, y en sus últimas versiones ofrece soporte para CSS3 y HTML5.

Por su parte, Safari y Google Chrome comparten un punto en común en su interior. Ambos navegadores utilizan **WebKit**, que en realidad es un framework que trabaja en el corazón de ambos navegadores y también de muchas otras soluciones, como por ejemplo, dispositivos móviles y tablets. WebKit incluso es utilizado por productos relacionados con el desarrollo web, como es el caso de Adobe Dreamweaver.

Como ya hemos visto antes, Presto es el motor de renderizado correspondiente al navegador web Opera desde su versión 7 y resulta ser uno de los que mejores prestaciones ofrece en la actualidad, en cuanto a compatibilidad con el nuevo HTML5.

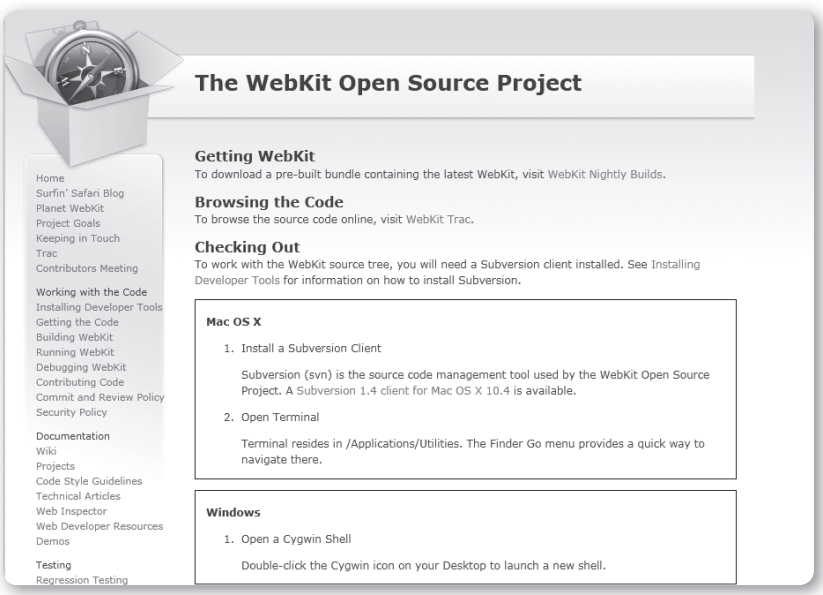


Figura 24. WebKit es Open Source, y su código se puede descargar ingresando en **`http://webkit.org/building/checkout.html`**.

Herramientas de desarrollo

Debemos tener en cuenta que tanto el lenguaje HTML como CSS y también JavaScript son lenguajes que pueden escribirse en cualquier editor de textos, y no necesitan compilación.

Por otro lado, las claras ventajas que ofrecen algunas herramientas de desarrollo son las siguientes: la identificación del código con diferentes colores, sistema de ayudas en la sintaxis y también otras facilidades relacionadas con la construcción de código.

Software WYSIWYG

What You See Is What You Get (**WYSIWYG**) es un término en inglés que suele traducirse como: lo que ves es lo que obtienes. Las aplicaciones que adhieren a este concepto ofrecen un entorno de desarrollo que, además de mostrar el código, permiten ver lo que ocurre de una manera visual, previsualizando el resultado.

Figura 25. Dreamweaver es una herramienta paga, existe una versión de prueba en **`www.adobe.com/es/products/dreamweaver`**.

Entre los editores WYSIWYG para lenguajes y tecnologías web, sin duda el más famoso es **Dreamweaver**. Lanzado originalmente por la empresa Macromedia (en el año 1997 para Mac y posteriormente en 1998 para plataformas basadas en PC), en la actualidad es un producto que forma parte de la Creative Suite de la empresa Adobe.

Se destaca por ofrecer vista de diseño y vista de código. Brinda soporte para HTML, CSS, JavaScript y PHP, entre otros lenguajes.

Para trabajar con Dreamweaver, podemos combinar las vistas de diseño y código o manejarnos sobre una de ellas en particular. En este aspecto, el programa nos ofrece diferenciación por colores en el marcado y ayuda contextual al escribir en los lenguajes soportados. Si trabajamos directamente sobre el código, podremos tener mayor control sobre el desarrollo que estemos realizando.

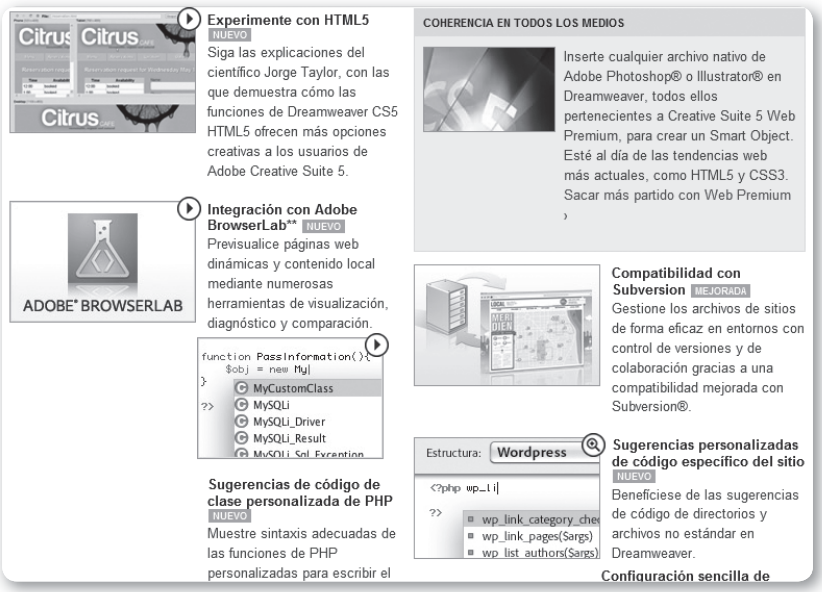


Figura 26. Dreamweaver fue pensado para diseñadores web, pero también es útil para los que trabajan en forma directa sobre el código.

En 2010, Adobe lanzó su Creative Suite 5 y, también, comenzó a publicar packs relacionados con HTML5 y CSS3 para sus productos. Más tarde, se lo incluyó en Dreamweaver CS5 11.0.3 Updater.

Con la llegada de la versión CS 5.5, las características de HTML5 y CSS3 se encuentran integradas perfectamente en la aplicación y se pueden utilizar de manera mucho más eficiente.

Otras alternativas

En el ambiente de desarrollo web, goza de gran prestigio una herramienta muy simple, liviana y efectiva. **Notepad++** es una aplicación gratuita (su licencia es GNU) desarrollada para Windows, que brinda la posibilidad de escribir código fuente de una gran variedad de lenguajes, entre los que podemos mencionar HTML, XML, CSS, JavaScript, PHP, Ruby, Perl y SQL, entre otros.

Entre las ventajas para destacar encontramos: marcado en color de la sintaxis, envoltura de sintaxis, niveles de zoom, soporte para macros y la posibilidad de utilizar extensiones.

Disponible en diferentes idiomas, incluso en español, su enlace de descarga se encuentra en **http://notepad-plus-plus.org/download**.

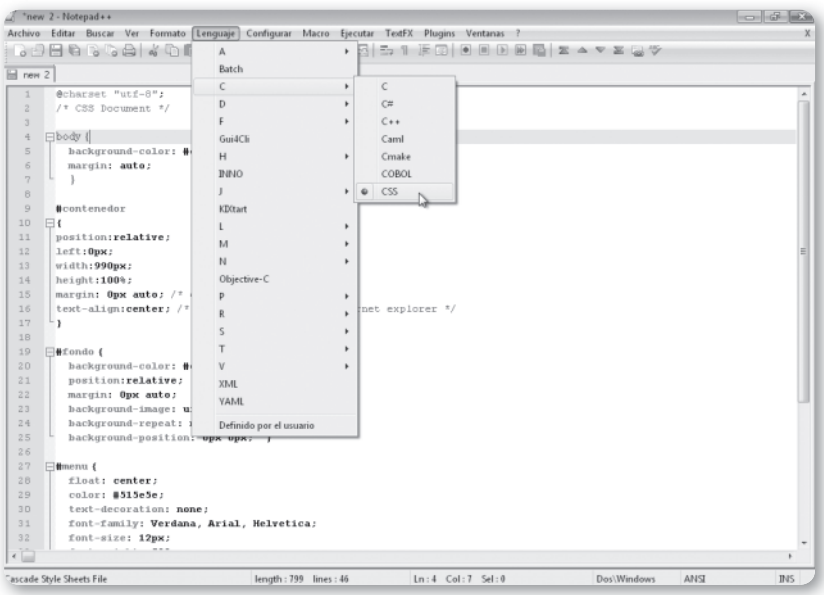


Figura 27. Notepad++ se destaca por la simpleza y flexibilidad que ofrece para escribir código y definir el lenguaje de programación.

Es interesante tener en cuenta que una alternativa multiplataforma muy funcional es la que ofrece el entorno de desarrollo denominado **Eclipse**. Este funcional proyecto se encarga de ofrecer un IDE abierto y extensible para que los programadores puedan aprovechar su flexibilidad y características en los proyectos que deseen.

Utilizado en especial por desarrolladores de lenguaje Java, debemos saber que Eclipse también ofrece muy buenas alternativas para quienes deseen programar utilizando JavaScript y PHP.

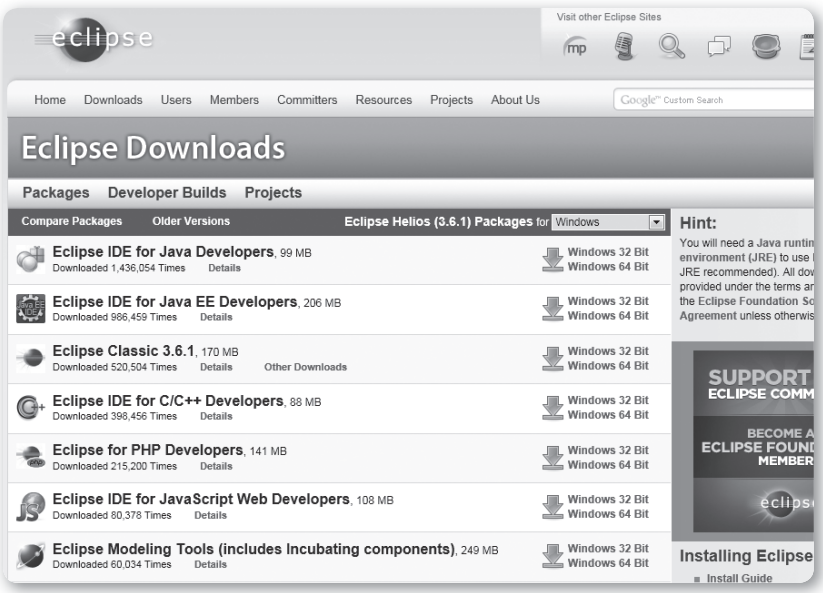


Figura 28. En **`www.eclipse.org/downloads`** encontraremos las opciones para descargar las herramientas de desarrollo.

MICROSOFT Y EL DISEÑO WEB

Frontpage fue el editor WYSIWYG de páginas web que formó parte de las suites de Microsoft Office en las versiones lanzadas entre los años 1997 y 2003. En la actualidad, Microsoft ofrece como alternativas para diseño de páginas web las aplicaciones llamadas **SharePoint Designer** (distribuida con una licencia freeware) y **Expression Web** (parte de la suite comercial Expression Studio).

Inspección de código desde el navegador

Muchos navegadores han incorporado, de manera nativa o a través de plugins y agregados, la posibilidad de **inspeccionar el código** de los sitios web en el cual estamos navegando.

Google Chrome ofrece, de manera nativa, uno de los mejores inspectores de código accesibles desde el navegador.

Figura 29. La inspección de código de Chrome también nos permite realizar modificaciones sobre las propiedades.

Para utilizar esta herramienta, es necesario navegar el sitio; en la página elegida, hacemos clic derecho sobre el elemento que deseamos inspeccionar y, en el desplegable, elegimos **Inspeccionar elemento**. De esta forma se abrirá una ventana que se encarga de mostrarnos el código de la página y las propiedades del elemento.

El navegador web de Microsoft, Internet Explorer, nos brinda una opción llamada **Herramienta de Desarrollo**, a la que se puede acceder mediante la tecla **F12**. Entre sus posibilidades, ofrece vista del

código HTML, CSS, Script, opciones de búsqueda y también validación, entre otras interesantes y útiles opciones.

Por su parte, Firefox cuenta con un complemento muy difundido entre los desarrolladores web. Su nombre es **Firebug** y, entre sus características principales para inspección de código, destacamos la posibilidad de modificaciones de estilos en tiempo real, debugger para JavaScript, visualización de métricas de CSS, monitoreo de la actividad en la red y exploración de DOM, entre muchas otras ventajas.

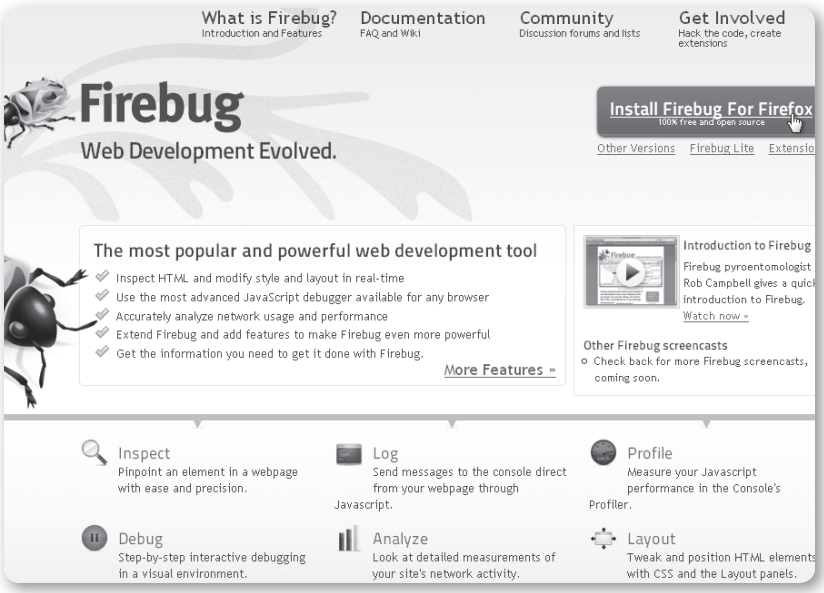


Figura 30. Firebug es gratuito y se puede obtener en el sitio web que se encuentra en la dirección **`http://getfirebug.com`**.

PROGRAMACIÓN ORIENTADA A OBJETOS

Es interesante tener en cuenta que la técnica de programación que se basa en el paradigma de objetos comenzó a hacerse de un importante lugar a partir de la década del noventa. Varios lenguajes utilizados para desarrollo web, como por ejemplo PHP, comenzaron a incorporar el manejo de objetos para ofrecer a los desarrolladores mayor potencial en las técnicas de programación.

Test de compatibilidad

Para conocer el nivel de compatibilidad de nuestros sitios, además de probarlos en los diferentes navegadores, también es posible usar tests de compatibilidad. Así podremos tener más herramientas para establecer el nivel de compatibilidad frente a los estándares web.

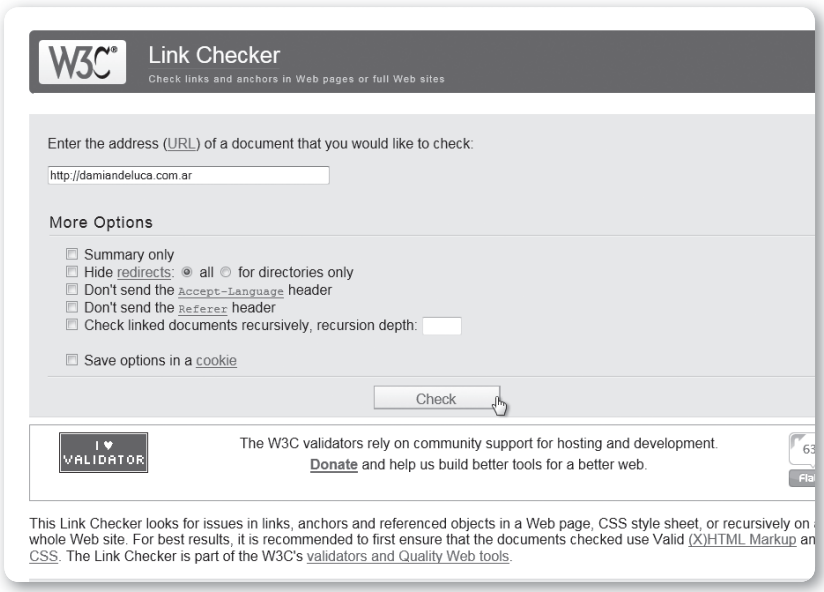


Figura 31. Es muy útil la herramienta de validación de links del W3C, `Link Checker` (`http://validator.w3.org/checklink`).

El W3C cuenta con varias herramientas de verificación, entre ellas, **Markup Validation Service** (**http://validator.w3.org**), un validador

LOS IDIOMAS MÁS UTILIZADOS EN INTERNET

Algunos pueden considerar a Internet como una gran Torre de Babel moderna. Más allá de esta reminiscencia bíblica, está claro que el lenguaje "universal" de Internet es el inglés. Casi a la par, aparece el chino y, varios escalones más abajo, en cuanto a cantidad de contenidos, se ubican el español, el japonés y el alemán.

de código HTML y XHTML; y **CSS Validation Service** (**http://jigsaw.w3.org/css-validator**), que realiza validación sobre las hojas de estilo.

Por su parte, **Acid3** es uno de los tests de compatibilidad más reconocidos hoy en día. Para realizar la evaluación, simplemente debemos ingresar con el navegador que deseamos testear en el sitio web que encontramos en la dirección **http://acid3.acidtests.org**.

Al acceder al sitio, de forma automática, correrá el test, que consiste en 100 pruebas que se ejecutarán una tras otra, y entregarán la puntuación final en pantalla. Entre las características que se verifican en esta prueba se encuentran: elementos HTML, selectores CSS3, acceso a DOM y también SVG, entre otras opciones.

Técnicas de detección de compatibilidad

Existen diversas técnicas para detectar el navegador y sus características. Mediante JavaScript, es posible saber qué navegador utiliza el usuario que llega a nuestro sitio y, de esta manera, definir qué vamos a mostrar según el caso. Si bien lo ideal es construir siempre sitios y aplicaciones web cross-browser (que funcionen en todos los navegadores), la decisión de utilizar lenguajes y tecnologías modernas, como puede ocurrir con HTML5 y CSS3, pueden causar efectos no deseados en navegadores antiguos o no compatibles.

Para simplificar nuestro trabajo, es posible recurrir a librerías de JavaScript específicas para esta finalidad.

En lo que se refiere a HTML5 y CSS3, una librería muy interesante es **Modernizr** (**www.modernizr.com**). Este framework de JavaScript se destaca por ser muy liviano y permitir que se realice la fácil detección de la compatibilidad nativa que ofrece el navegador respecto de las tecnologías de última generación que se están utilizando.

Esta librería nos permite utilizar HTML5 y también CSS3 en nuestros proyectos web y de esta forma detectar los navegadores que no ofrecen compatibilidad con alguna de sus características, para que podamos aplicar una solución alternativa en estos casos. El script se encarga de agregar una clase a los elementos HTML, de acuerdo con las

características del navegador. Cabe destacar que Modernizr no es una herramienta para agregar nuevas funcionalidades al navegador, sino que es un framework que nos permite saber las limitaciones del software con el que un cliente visita nuestro sitio y actuar en consecuencia, de esta forma estaremos preparados.

Figura 32. Modernizr es una alternativa eficiente para utilizar las novedades de HTML5 y CSS3, y resolver problemas de compatibilidad.

RESUMEN

En este primer capítulo, hemos explorado la evolución de la Web, desde lo referente a tendencias, tecnologías y cambios culturales. Vimos la importancia de la estandarización de formatos y el rol del W3C en Internet. Analizamos qué son y para qué sirven los lenguajes de etiquetas y trazamos la evolución de HTML hasta la llegada de HTML5. Dentro de los lenguajes y tecnologías que interactúan con HTML destacamos la importancia de CSS y AJAX. Vimos las técnicas más importantes para el desarrollo web, describimos los navegadores más relevantes del mercado y, para finalizar, nos encargamos de analizar algunas interesantes técnicas que nos ayudarán en la tarea de detectar compatibilidad.

Actividades

TEST DE AUTOEVALUACIÓN

1	Indique qué diferencias hay entre Web 1.0, 2.0 y 3.0.
2	¿Qué es el W3C?
3	¿Qué características tiene un lenguaje de etiquetas?
4	¿Qué diferencias existen entre HTML4 y XHTML?
5	¿Cuáles son las ventajas que introduce HTML5?
6	¿Para qué sirve utilizar CSS en un proyecto web?
7	¿Para qué se utiliza JavaScript?
8	¿Qué ventajas introduce el uso de AJAX?
9	Defina qué es DOM.
10	¿Cuál es la función del motor de renderizado de un navegador web?

ACTIVIDADES PRÁCTICAS

1	Descargue Notepad++ en su equipo, instálelo y cree un documento XML.
2	Visualice una página web con Google Chrome y verifique los elementos con el Inspector de Elementos.
3	Verifique la compatibilidad de su sitio con Acid3
4	Realice una verificación de su sitio con el validador del W3C.
5	Descargue Modernizr e inclúyalo en su proyecto web.

*02 –

Novedades de HTML5

En el Capítulo 2, vamos a realizar una recorrida que nos brindará la posibilidad de conocer para qué sirven los elementos del lenguaje HTML4 y de XHTML. También haremos un análisis que nos mostrará la evolución de sus características con la llegada de HTML5. Veremos qué elementos se retiran del estándar, los que reciben algún tipo de cambio y, por supuesto, los que se incorporan a partir de HTML5.

- **Qué son los elementos HTML 64**
  - Los atributos en HTML 64
  - Los eventos en HTML..... 66
- **Elementos HTML4/XHTML 68**
- **Elementos que se retiran del estándar o cambian con HTML5 ..... 86**
  - Atributos que dejan de ser soportados..... 87
  - Elementos que cambian con HTML5 88
- **Características que incorpora HTML5 ..... 89**
  - Atributos en HTML5 91
  - Los eventos que se incorporan con HTML5 92
- **Resumen..... 97**
- **Actividades..... 98**

Qué son los elementos HTML

HTML es un lenguaje que se compone por elementos que permiten definir la estructura del documento. Estos elementos son los que nos posibilitan determinar cómo estará armada la página y sus secciones.

Las etiquetas nos brindan la oportunidad de definir los elementos en el código. Como hemos visto en el capítulo anterior, no todas las etiquetas se utilizan para representación, ya que también pueden tener otra finalidad (por ejemplo, para ofrecer información del documento).

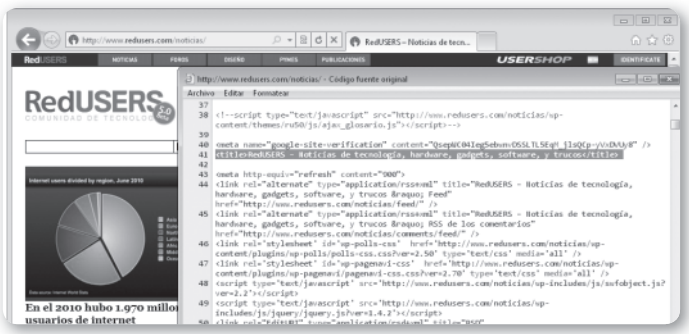


Figura 1. El contenido de la etiqueta **`<title>`** ofrece esta información en la barra de título o en las pestañas del navegador.

Los atributos en HTML

Al introducirnos en el lenguaje, es importante saber que los atributos en HTML se emplean para poder especificar ciertas características que tendrá el elemento o bien para ofrecer más información sobre él.

Existen atributos que son obligatorios y otros, opcionales. Por ejemplo, si utilizamos la etiqueta de imagen, será necesario indicar la dirección de la cual se debe obtener, pero no resulta obligatorio indicar si tiene un texto alternativo, ya que este atributo puede estar vacío.

Es importante tener en cuenta que hay atributos globales, que son soportados por la mayoría de los elementos (como **id**, **class**, **style**, etcétera) y otros que son muy particulares del elemento (como pueden ser algunos atributos de tablas o de controles de formulario).

A continuación, veremos los principales atributos de HTML:

- **id**: identificador único del elemento (no pueden existir dos elementos con la misma **id** en un mismo documento).
- **class**: permite indicar el nombre de la clase del elemento.
- **style**: se usa para especificar el estilo (inline) para el elemento.
- **title**: hace posible escribir información adicional sobre el elemento.
- **dir**: permite especificar la dirección para el contenido en el elemento (el valor puede ser **rtl** o **ltr**).
- **lang**: se utiliza para especificar el código del lenguaje para el contenido del elemento. Para español se usa **lang="es-ES"**.
- **xml:lang**: permite definir el código de lenguaje para elementos (en XHTML). El ejemplo para español es: **xml:lang="es"**.
- **charset**: permite especificar la codificación de caracteres del documento al que se hace referencia. Por lo general se utiliza UTF-8 (para unicode) o ISO-8859-1 (para alfabeto latino).
- **media**: para especificar el tipo de dispositivo de salida. Su valor puede ser **screen**, **tty**, **tv**, **projection**, **handheld**, **print**, **braille**, **aural**, **all**.
- **accesskey**: tecla de acceso rápido.
- **href**: permite indicar la URL de un archivo o recurso.
- **hre**f **ang**: permite indicar el lenguaje del texto en el documento.
- **rel**: se encarga de indicar la relación entre el documento actual, respecto del que se está vinculado.
- **rev**: indica la relación entre el documento vinculado y el actual.
- **target**: permite indicar el destino para el documento vinculado. Puede ser: **_blank**, **_self**, **_top** y **_parent**.
- **type**: permite especificar el tipo MIME.
- **align**: permite indicar la alineación del elemento. Puede tomar los valores **left**, **center**, **right** o **justify**. Es un atributo que, desde hace tiempo, ha dejado de utilizarse ya que esto debe manejarse de CSS.
- **height**: define el alto del elemento.
- **width**: se usa para definir el ancho del elemento.

DIRECCIONES URL RELATIVAS VS. ABSOLUTAS

Al indicar la dirección de un recurso, podemos especificar su ubicación de manera relativa o absoluta. Hacemos referencia de manera relativa a un archivo, cuando indicamos la ruta a partir del archivo que lo está llamando. Una URL absoluta se da cuando se indica la ruta completa al recurso.

Figura 2. Podemos definir el destino para un recurso, pero el usuario puede elegir dónde lo abrirá gracias a las opciones del navegador.

En el apartado **Elementos HTML4/XHTML**, de este mismo capítulo, veremos qué atributos soporta cada elemento HTML/XHTML. Allí también analizaremos otros atributos específicos para algunos elementos.

Si deseamos saber más, la tabla completa de atributos para HTML4 que brinda el W3C la encontraremos en el sitio web que está en la dirección **www.w3.org/TR/html4/index/attributes.html**.

Los eventos en HTML

Una característica interesante de HTML tiene que ver con la posibilidad de interactuar con eventos dinámicos que pueden producirse por diferentes tipos de interacción, ya sea por parte del usuario o por parte del propio documento.

Así como los elementos HTML soportan atributos, también pueden tener asignados eventos, que pueden trabajar junto con un script del lado cliente. Cuando un evento ocurre, una acción se ejecuta. Por lo general, correrá un código JavaScript que definimos para dicho fin.

Para comprender mejor los eventos de HTML, podemos separarlos básicamente en cinco grupos: de documento, de imagen, de formulario, del teclado y del mouse. A continuación veremos los principales.

Los eventos del documento son:

- **onload**: ocurre cuando termina de cargar el documento.
- **onunload**: se trata de un evento que ocurre cuando el documento es quitado de la ventana por parte del usuario.

Para imagen tenemos:

- **onabort**: ocurre cuando la carga de la imagen es abortada.

Los eventos para formularios son:

- **onselect**: este interesante evento ocurre cuando se selecciona el contenido del elemento o control correspondiente.
- **onchange**: ocurre cuando el elemento del formulario cambia su valor.
- **onfocus**: ocurre cuando el elemento recibe el foco.
- **onblur**: ocurre cuando el elemento pierde el foco.
- **onreset**: se produce cuando se resetea el formulario.
- **onsubmit**: se produce cuando el formulario es enviado.

Los eventos del teclado son:

- **onkeypress**: se trata de un evento que ocurre cuando una tecla es presionada y soltada por el usuario.
- **onkeydown**: ocurre cuando una tecla es pulsada por el usuario.
- **onkeyup**: ocurre cuando el usuario libera la tecla que tenía pulsada.

Los eventos del mouse son:

- **onmousemove**: ocurre cuando el mouse se mueve sobre el elemento.
- **onmouseout**: ocurre cuando el mouse es movido fuera del elemento.
- **onmouseover**: ocurre cuando el mouse se mueve desde afuera y pasa por encima del elemento en cuestión.
- **onclick**: este evento ocurre cuando el usuario se encarga de hacer clic con el botón del mouse sobre el elemento.
- **ondblclick**: se trata de un evento que ocurre cuando el usuario procede a hacer doble clic con el botón del mouse.
- **onmousedown**: ocurre cuando el botón del mouse es presionado.
- **onmouseup**: ocurre cuando soltamos el botón del mouse.

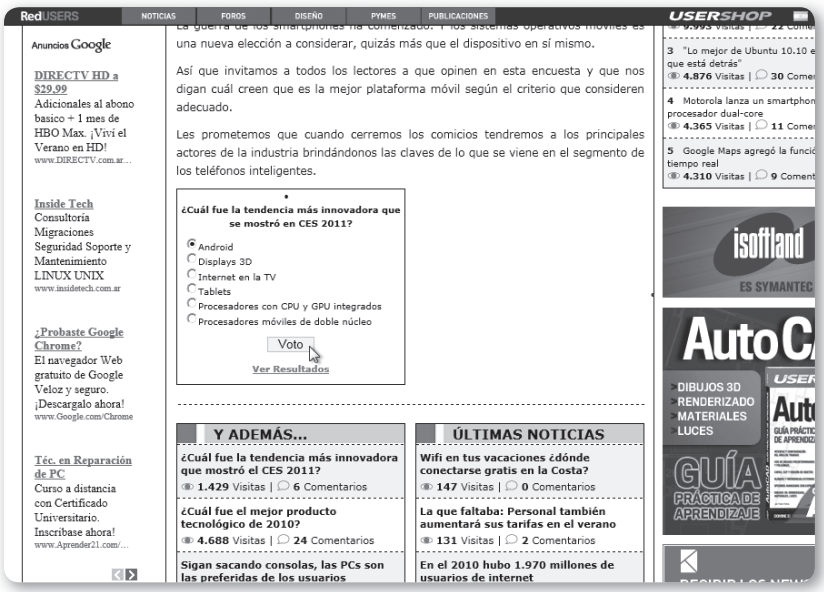


Figura 3. El evento **onclick** se puede utilizar, por ejemplo, para capturar el evento del botón de un formulario.

Los que hemos nombrado son eventos compatibles con HTML4 y XHTML. Con la llegada de HTML5 se incorporan nuevos eventos para potenciar las posibilidades de interacción con este lenguaje.

Elementos HTML4/XHTML

Ahora veremos las etiquetas más importantes de HTML4 y XHTML, junto a su descripción y los principales atributos soportados. Aquellos atributos que se indican como no recomendados o desaconsejados pueden ser reemplazados por otras técnicas, generalmente relacionadas con representación mediante CSS.

En la **Tabla 1**, encontramos etiquetas del tipo de documento y del encabezado. Vale aclarar que algunas de ellas también pueden usarse en el cuerpo (por ejemplo, **<script>** o **<style>**), aunque es recomendable que se ubiquen en la cabecera del documento..

ETIQUETAS PARA ENCABEZADO

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<!DOCTYPE>	Se emplea para especificar el tipo de documento.	No posee.
<html>	Define y establece el inicio en un documento HTML.	**xmlns** (solo para XHTML), **dir**, **lang** y **xml:lang**.
<head>	Indica el comienzo del encabezado del documento.	**lang**, **xml:lang**, **dir** y **profile** (URL del documento con metadata de perfiles)
<title>	Permite indicar el título del documento. Se ubica en el encabezado del documento.	**lang**, **dir**, **xml:lang**.
<meta>	Con esta etiqueta, es posible incluir información de metadatos en el documento. Por ejemplo, descripción, palabras clave, autor, etcétera.	**content** (es requerido para definir el contenido de la metaetiqueta), **http-equiv** (encabezado HTTP para información del contenido del atributo), **name** (provee de nombre a la información del atributo: **description**, **author**, **keywords**, **generator**, etcétera), **scheme** (contiene el formato o la URL con la información del esquema que debe utilizarse para interpretar el contenido del atributo), **dir**, **lang** y **xml:lang**.
<link>	Se utiliza para vincular archivos externos. Define una relación entre el documento actual y una fuente externa (por ejemplo, un archivo CSS). Se ubica en el encabezado del documento.	**charset**, **href**, **hre**f **ang**, **media**, **rel**, **rev**, **target**, **type**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<style>	Se utiliza para definir los estilos CSS del documento.	**type**, **media**, **dir**, **title**, **lang** y **xml:lang**.
<script>	Contiene los scripts del lado cliente del documento. Si se utiliza el atributo **src**, es posible indicar la ruta de un archivo de script externo.	**type**, **charset**, **src**, **defer** (permite indicar que la ejecución del script se demore hasta que cargue la página).

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<noscript>	Permite definir un contenido alternativo para los navegadores que no soportan scripts del lado cliente.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

Tabla 1. En esta tabla se muestran las etiquetas HTML más utilizadas en el encabezado del documento, así como sus principales atributos.

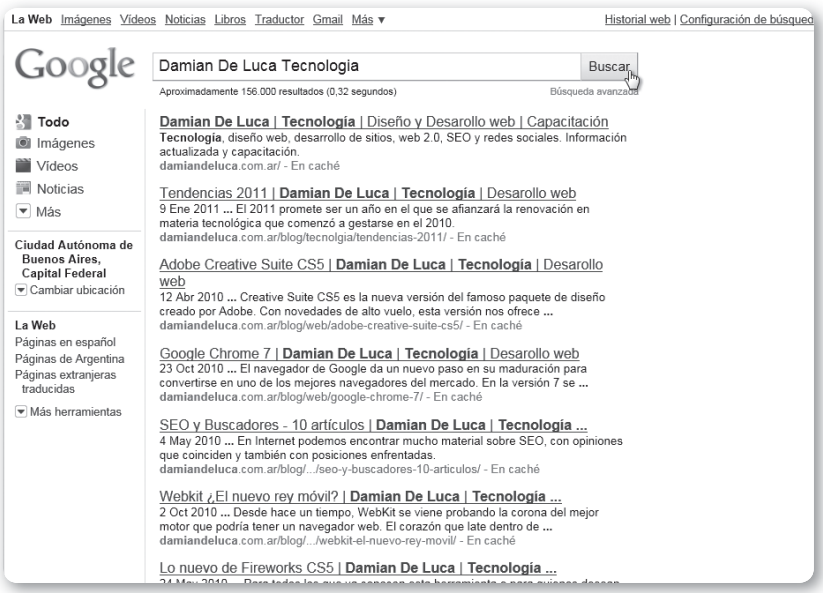


Figura 4. Al emplear **<meta>** para indicar la descripción de una web, de cara a los buscadores, se recomienda utilizar textos específicos.

ENTIDADES HTML

Cuando se necesita incluir algún carácter especial (letras con acento o diéresis, símbolos, etcétera) en el texto en HTML, es conveniente convertirlo en entidad para asegurarnos de que todos los navegadores lo interpreten de manera correcta. Encontraremos más información y tablas que nos ayudarán a realizar la conversión en **www.w3schools.com/tags/ref_entities.asp**.

A partir de aquí, analizaremos las etiquetas relacionadas con el cuerpo del documento. En la tabla que veremos a continuación, se incluyen las etiquetas de declaración del cuerpo, distintas categorías de títulos, párrafos, bloques de texto, citas, código de computadoras, saltos de línea, reglas horizontales, enlaces e imágenes.

ETIQUETAS QUE SE USAN EN EL CUERPO DEL DOCUMENTO

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<body>	Marca el comienzo del cuerpo del documento.	**link** (estilo del color de los enlaces visitados), **alink** (estilo de los enlaces activos), **vlink** (estilo del color del enlace visitado), **background** (estilo del fondo), **bgcolor** (estilo del color de fondo; al igual que los anteriores atributos, no es recomendado su uso), **text** (estilo del color de texto), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<h1> **<h2>** **<h3>** **<h4>** **<h5>** **<h6>**	Se utilizan para definir los distintos niveles de título de un documento HTML. **<h1>** es el de mayor jerarquía y **<h6>** es el de menor.	**align** (no es recomendado su uso), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<p>	Se utiliza para definir un párrafo.	**align** (no es recomendado su uso), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<pre>	Se usa para texto preformateado.	**width** (máximo de caracteres por línea, no es recomendado su uso), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<blockquote>	Se emplea para indicar una cita larga (un bloque).	**cite** (permite especificar la fuente de la cita), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<q>	Se utiliza para citas cortas.	**cite** (se trata de un atributo que nos permite especificar la URL de la fuente de la cita correspondiente), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<cite>	Permite definir una cita (o referencia).	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<code>	Se emplea para indicar que es un texto de computadora.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<samp>	Se utiliza para ejemplificar un texto de código de computadora.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
** ** (HTML4) ** ** (XHTML)	Inserta una línea en blanco en el f ujo (como si fuera un **ENTER** en el texto).	**id**, **class**, **style** y **title**.
<hr> (HTML4) **<hr />** (XHTML)	Se emplea para crear una línea horizontal.	**noshade** (el elemento se debe representar en un color sólido), **size** (el alto del elemento), **width** (ancho del elemento), **align** (al igual que los anteriores atributos, su uso está desaconsejado), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<a>	Se utiliza para marcar un enlace o ancla (hacia afuera o hacia dentro del documento HTML).	**charset**, **href**, **hre**f **ang**, **coords** (coordenadas del enlace), **shape** (forma del enlace), **name** (nombre del ancla), **target**, **rel**, **rev**, **accesskey**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<base>	Permite definir la dirección o el destino por defecto en los links de una página.	**href**, **target**.
<img>	Se utiliza para definir una imagen.	**align**, **border**, **height**, **hspace**, **ismap**, **longdesc**, **usemap**, **vspace** (desaconsejado), **width**, **alt** (texto alternativo para la imagen), **src**, **id**, **class**, **dir**, **style**, **title**, **lang** y también **xml:lang**.

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<map>	Permite definir un mapa de imagen (image-map).	**name** (nombre del mapa de imagen), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<area>	Si la utilizamos con un mapa de imágenes nos permite definir áreas que deseemos.	**alt**, **accesskey**, **href**, **nohref** (para indicar que no tiene link), **shape** (para especificar la forma del área), **coords** (coordenadas), **target**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

Tabla 2. Etiquetas HTML adecuadas para marcar el comienzo del cuerpo del documento, títulos, párrafos, bloques de texto, citas, código de computadoras, saltos de línea, reglas horizontales, enlaces e imágenes.

Figura 5. El atributo **`title`** dentro de la etiqueta **`<img>`** es útil para que se muestre el texto especificado al pasar el mouse por encima.

En la tabla que se presenta más abajo, encontraremos los elementos relacionados con las tablas HTML. Entre otras, veremos etiquetas para definir una tabla completa o una celda, entre otras opciones.

ELEMENTOS PARA DEFINIR UNA TABLA

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<table>	Se emplea para definir una tabla.	**align** (desaconsejado), **bgcolor** (color de fondo, desaconsejado), **border** (ancho del borde), **cellpadding** (espacio entre la celda y su contenido), **cellspacing** (espacio entre las celdas), **frame** (especifica las partes de afuera de los bordes que serán visibles), **rules** (especifica las partes interiores de los bordes que serán visibles), **summary** (sumario de contenido de la tabla), **width** (ancho de la tabla), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<td>	Con esta etiqueta se define una celda en una tabla.	**abbr** (texto abreviado del contenido de la celda), **align**, **axis** (categoría de la celda), **bgcolor** (color de fondo de la celda, desaconsejado), **char** (alinea el contenido de la celda a un carácter), **charoff** (permite especificar el número de caracteres para alinear, respecto de los indicados en **char**), **colspan** (permite indicar el número de columnas que la celda puede abarcar), **headers** (encabezado de la tabla relativo a una celda), **scope** (indica la manera de asociar los encabezados de celda con el contenido de celda en la tabla), **height** (desaconsejado), **width**, **nowrap** (indica que el contenido no se puede envolver, desaconsejado), **rowspan** (indica el número de celdas que pueden ser abarcadas), **valign** (alineación vertical en la celda, desaconsejado), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

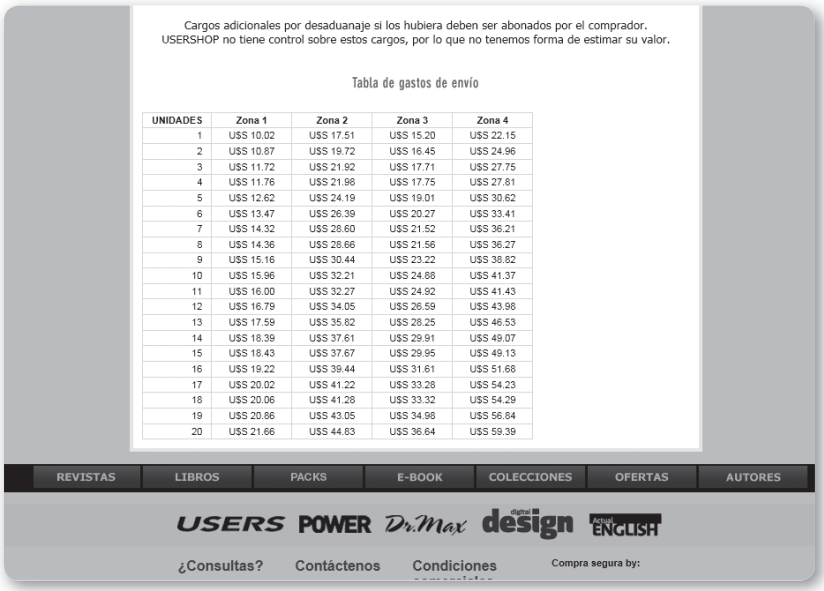
ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<tr>	Define una fila en una tabla.	**align**, **bgcolor** (desaconsejado), **char**, **charoff**, **valign**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<th>	Se utiliza para definir la celda de encabezado de una tabla.	**abbr**, **align**, **axis**, **bgcolor** (desaconsejado), **char**, **charoff**, **colspan**, **scope**, **height** (desaconsejado), **width** (desaconsejado), **nowrap** (desaconsejado), **rowspan**, **valign**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<col>	Permite indicar los atributos de una columna de una tabla.	**align**, **char**, **charoff**, **span**, **valign**, **width**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<colgroup>	Se utiliza para agrupar columnas de una tabla (para poder darles formato).	**align**, **char**, **charoff**, **span**, **valign**, **width**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<tbody>	Agrupa contenido en el cuerpo de una tabla.	**align**, **char**, **charoff**, **valign**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<tfoot>	Define el pie de una tabla.	**align**, **char**, **charoff**, **valign**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<caption>	Permite definir el título de una tabla.	**align** (desaconsejado), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

Tabla 3. Elementos HTML que se utilizan para definir y estructurar una tabla.

RECOMENDACIONES PARA EL MAQUETADO

Al realizar el maquetado de una página web, no solo debemos pensar en la estética, sino también en la limpieza y la correcta utilización de los elementos HTML. Así como está desaconsejado maquetar con tablas y con frames, el uso recomendado de elementos para definir áreas (como por ejemplo el caso de ** **) tampoco debe ser utilizado en exceso, en especial al anidarlos.

Con las etiquetas vistas, no solo definimos una tabla y sus celdas, también podemos agrupar columnas y definir un título.



UNIDADES	Zona 1	Zona 2	Zona 3	Zona 4
1	U$S 10.02	U$S 17.51	U$S 15.20	U$S 22.15
2	U$S 10.87	U$S 19.72	U$S 16.45	U$S 24.96
3	U$S 11.72	U$S 21.92	U$S 17.71	U$S 27.75
4	U$S 11.76	U$S 21.98	U$S 17.75	U$S 27.81
5	U$S 12.62	U$S 24.19	U$S 19.01	U$S 30.62
6	U$S 13.47	U$S 26.39	U$S 20.27	U$S 33.41
7	U$S 14.32	U$S 28.60	U$S 21.52	U$S 36.21
8	U$S 14.36	U$S 28.66	U$S 21.56	U$S 36.27
9	U$S 15.16	U$S 30.44	U$S 23.22	U$S 38.82
10	U$S 15.96	U$S 32.21	U$S 24.88	U$S 41.37
11	U$S 16.00	U$S 32.27	U$S 24.92	U$S 41.43
12	U$S 16.79	U$S 34.05	U$S 26.59	U$S 43.98
13	U$S 17.59	U$S 35.82	U$S 28.25	U$S 46.53
14	U$S 18.39	U$S 37.61	U$S 29.91	U$S 49.07
15	U$S 18.43	U$S 37.67	U$S 29.95	U$S 49.13
16	U$S 19.22	U$S 39.44	U$S 31.61	U$S 51.68
17	U$S 20.02	U$S 41.22	U$S 33.28	U$S 54.23
18	U$S 20.06	U$S 41.28	U$S 33.32	U$S 54.29
19	U$S 20.86	U$S 43.05	U$S 34.98	U$S 56.84
20	U$S 21.66	U$S 44.83	U$S 36.64	U$S 59.39

Figura 6. En esta figura, vemos un correcto uso de las tablas con la finalidad para la que han sido creadas.

En la tabla siguiente, nos encargaremos de analizar los elementos HTML para áreas de contenidos de marcos.

ELEMENTOS PARA ÁREAS DE CONTENIDOS Y FRAMES

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
** **	Permite envolver un área de contenido para definir una sección.	**align** (desaconsejado), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
****	Posibilita definir un área del documento. Se usa para poner referencias y asignar atributos o estilos.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<frameset>	Se trata de la etiqueta necesaria para que podamos definir un conjunto de marcos.	**cols** (columnas), **rows** (filas), **class**, **id**, **style** y **title**.
<frame>	Se usa para definir un marco.	**frameborder** (para indicar si el marco tiene borde o no, booleano), **longdesc** (descripción larga), **marginheight** (márgenes superiores e inferiores del marco), **marginwidth** (márgenes izquierdo y derecho), **noresize** (si el marco no puede cambiar de tamaño), **scrolling** (si el marco puede tener scroll), **scr**, **name**, **class**, **id**, **style** y **title**.
<noframes>	Permite definir un texto alternativo en los navegadores que no soportan frames.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<iframe>	Define un iframe, es decir, un marco incorporado o inline frame. Este marco puede mostrar otra página dentro del documento.	**align** (uso desaconsejado) **frameborder**, **height**, **longdesc**, **marginheight**, **marginwidth**, **scrolling**, **scr**, **name**, **class**, **id**, **style** y **title**.

Tabla 4. En esta tabla, se pueden ver elementos HTML que se emplean para definir aspectos tales como áreas de contenidos, y también aquellos relacionados con la definición y creación de frames.

En la tabla que se muestra en la siguiente página, nos encargaremos de realizar un análisis de cada uno de los elementos que están

NAVEGADOR LYNX

Lynx es el nombre de un navegador web de licencia GNU GPL. Su desarrollo nació en el año 1992 y, a lo largo de su historia, ofreció versiones para MS DOS, Microsoft Windows, OS/2, Linux y BSD, entre otros sistemas. Al ser un navegador que muestra solo texto, es útil para la verificación de usabilidad y accesibilidad de un sitio. El enlace a su sitio web es **http://lynx.isc.org**.

relacionados con formularios HTML. De esta forma, encontraremos etiquetas que nos permitirán indicar el inicio de un formulario, así como también agrupar los elementos que encontramos dentro de él, entre otras opciones relacionadas.

ETIQUETAS PARA FORMULARIOS

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<form>	Indica el comienzo de un formulario.	**action** (acción que realiza el formulario cuando es enviado), **accept** (tipos de archivos que pueden enviarse), **accept-charset** (juegos de caracteres aceptados), **enctype** (cómo debe ser codificada la información cuando es enviada al servidor), **method** (método de envío, puede ser **get** o **post**), **name**, **target** (desaconsejado) **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<input>	Define un control de entrada de un formulario.	**accept** (tipos de archivos aceptados), **align** (solo para tipo imagen, desaconsejado su uso), **alt**, **name**, **checked** (solo para radio o chekbox), **diseable**, **maxlength** (máximo de caracteres), **readonly** (solo lectura, solo para campos de texto o de contraseña), **size** (ancho del campo), **src**, **type** (tipo de input), **value** (valor), **accesskey**, **tabindex** (orden **tab** de los elementos), **accesskey**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<label>	Se utiliza para especificar el texto o etiqueta relacionado con un input.	**for** (indica a qué elemento del formulario está ligado), **accesskey**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<textarea>	Es un control de entrada de formulario para texto multilínea.	**cols**, **rows**, **disable**, **name**, **readonly**, **accesskey**, **tabindex**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<fieldset>	Permite agrupar un conjunto de elementos dentro de un formulario.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<legend>	Define la leyenda que lleva **<fieldset>**.	**align** (uso desaconsejado), **accesskey**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<select>	Permite definir una lista de selección (con posibilidad de ser desplegada)	**disabled**, **multiple**, **name**, **size**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<option>	Define una opción en una lista de selección.	**label**, **disabled**, **selected**, **value** (es el valor que será enviado por formulario), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<optgroup>	Posibilita definir un grupo de opciones relacionadas en una lista de selección.	**label** (texto con la descripción para el grupo de opciones), **disabled**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<button>	Se emplea para establecer un botón.	**disabled**, **name**, **type** (puede ser **button**, **reset** o **submit**), **value**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

Tabla 5. Elementos HTML vinculados con la creación de formularios.

LOS TIPOS PARA INPUT

La etiqueta **input** nos brinda la posibilidad de definir un control de entrada en un formulario. Una de sus características fundamentales, para establecer cómo se mostrará y también su función, es el atributo type. Entre los valores que puede recibir este atributo encontramos: button, checkbox, radio, file, hidden, image, password, reset, submit y text. En un mismo formulario se pueden utilizar uno o más controles de entrada, asignando a cada uno el tipo que corresponda a la función que deseemos que cumpla. De esta forma podemos darnos cuenta de que se trata de una etiqueta muy versátil y que debemos tener presente a la hora de trabajar en nuestros proyectos.

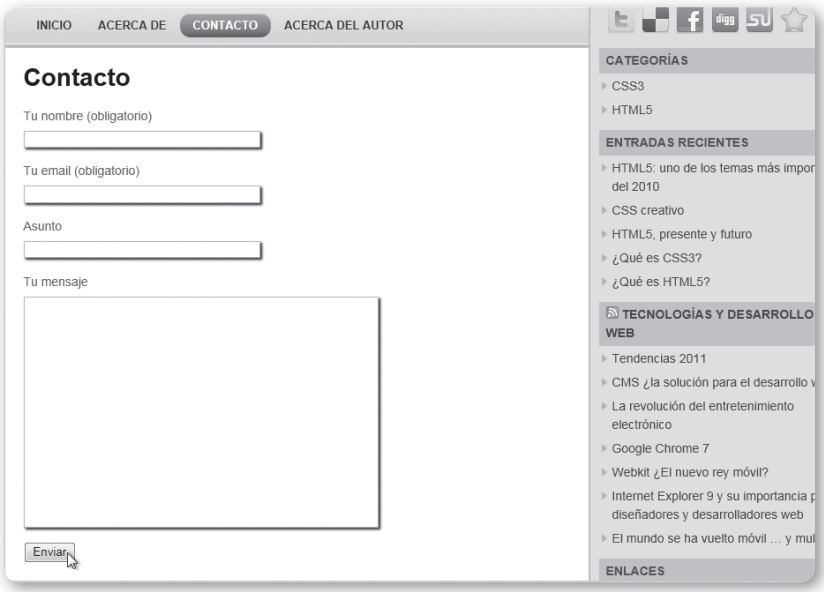


Figura 7. Los elementos del formulario se ven favorecidos con las ventajas que incorpora CSS3 para su representación.

La tabla que sigue a continuación nos mostrará los elementos relacionados con listas en HTML.

ELEMENTOS PARA LISTAS

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
** **	Permite definir una lista sin orden.	**compact** (para mostrar la lista compacta, su uso es desaconsejado), **type** (puede ser **disc**, **square** o **circle**; su uso es desaconsejado), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
** **	Se utiliza para definir una lista ordenada.	**compact** (para mostrarla compacta y pequeña, desaconsejado), **start** (punto de inicio de la lista), **type**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
** **	Define un ítem de una lista.	**type** (desaconsejado), **value** (desaconsejado), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<dl>	Se usa para crear una lista de definición.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<dt>	Su función es definir un término de la lista de definición.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<dd>	Permite definir la descripción de un término en una lista de definición.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<dfn>	Define un término de definición.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<menu> (uso desaconsejado para HTML4, se redefine en HTML5)	Permite definir una lista de menú.	**compact**, **id**, **class**, **dir**, **style**, **title**, **lang**.

Tabla 6. Elementos HTML que se utilizan para definir listas.

Las listas nos serán de gran utilidad para nuestros desarrollos, incluso mucho más de lo que algunos lectores puedan imaginar. Las listas nos pueden permitir definir desde menús desplegables y barras de navegación hasta bloques de anuncios.

CREAR UNA BARRA DE NAVEGACIÓN

La barra de navegación, también conocida como Navigation Bar o Navbar, nos permite definir un espacio en nuestra página donde ubicaremos un grupo de enlaces que le facilitarán la navegación al usuario. Las barras de navegación pueden estructurarse mediante listas HTML, definiendo luego su representación con el uso CSS. Los estilos nos permitirán definir la alineación, los colores y la orientación, entre otras características. Con CSS también es posible transformar los enlaces en botones, empleando diferentes técnicas. Las barras de navegación son muy útiles para potenciar la usabilidad y la navegabilidad del sitio y además son una solución ideal para mejorar la accesibilidad.

Figura 8. Las listas tienen mucha utilidad en la creación de sitios. Mediante CSS, podemos cambiar la forma en que se representan.

En la tabla que se presenta a continuación tendremos la oportunidad de analizar aquellas etiquetas que se encuentran relacionadas con fuentes, diferentes formas de representación de texto (tamaño, negritas, subrayados, etcétera) y también lo referente a acrónimos, abreviaturas, teletipo y texto del teclado, entre otras características de HTML muy interesantes relacionadas con el texto.

ELEMENTOS RELACIONADOS CON FUENTES Y TEXTOS

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<font> (uso desaconsejado)	Posibilita especificar algunas características de representación de la fuente, como color y tamaño.	**compact**, **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<sub>	Permite definir un subíndice.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<sup>	Posibilita indicar un superíndice.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<i> (se recomienda **** en su lugar)	Con esta etiqueta, se estable un texto en itálica (cursiva).	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<u> (uso desaconsejado)	Indica el subrayado del texto.	**id**, **class**, **dir**, **style**, **title**, **lang**.
<strike> y **<s>** (uso desaconsejado)	Se utilizan para mostrar texto tachado.	**id**, **class**, **dir**, **style**, **title**, **lang**.
<b> (se recomienda **** en su lugar)	Permite establecer una negrita en el texto.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
****	Permite destacar con énfasis.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
****	Asigna un énfasis fuerte a un texto.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<big> (uso desaconsejado)	Indica que el texto debe ir en tamaño grande.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<small> (uso desaconsejado)	Define un texto cuyo tamaño es pequeño.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
** ** (uso desaconsejado)	Indica que el texto se muestra centrado.	**id**, **class**, **dir**, **style**, **title**, **lang**.
<ins>	Indica un texto insertado.	**cite** (URL del documento que explica por qué el texto ha sido insertado o cambiado), **datatime** (fecha en que el texto fue insertado o cambiado), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<del>	Se emplea para indicar un texto borrado.	**cite** (URL del documento que explica por qué el texto ha sido borrado), **datatime** (fecha en que el texto fue borrado), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<bdo>	Permite definir la dirección del texto.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**. Aclaración: en este caso **dir** es obligatorio.
<dir> (uso desaconsejado)	Permite definir una lista de directorio (un árbol).	**compat**, **id**, **class**, **dir**, **style**, **title**, **lang**.

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<abbr>	Se utiliza para indicar abreviaturas.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<basefont > (uso desaconsejado)	Permite definir, de manera predeterminada, una fuente, su color y su tamaño. Solo soportado por Internet Explorer.	**color**, **face**, **size**, **id**, **class**, **dir**, **style**, **title** y **lang**.
<acronym>	Se emplea para definir acrónimos.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<tt>	Texto de teletipo.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<kbd>	Define el texto del teclado.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<address>	Permite especificar la información de un contacto.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

Tabla 7. Elementos HTML que se utilizan para definir listas.

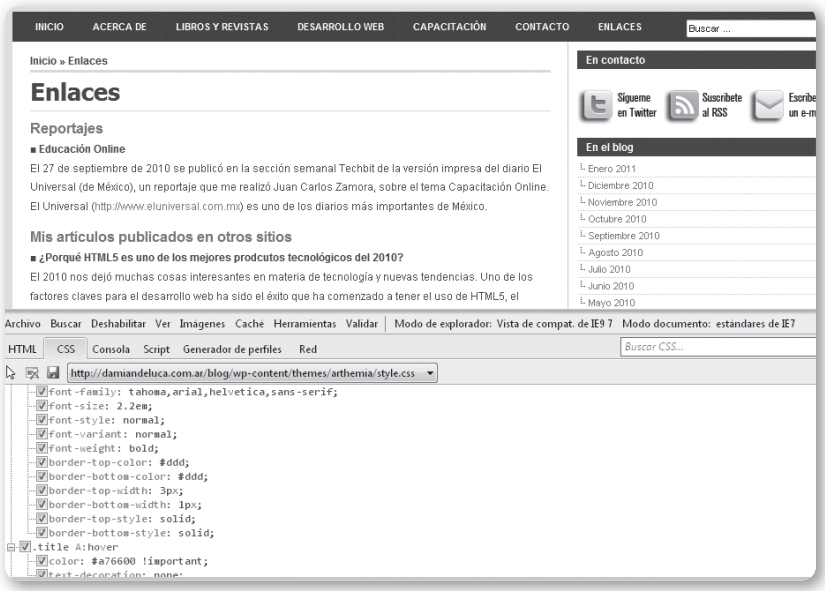


Figura 9. Las características de representación deben ser asignadas desde CSS, no es recomendable hacerlo mediante HTML.

En la próxima tabla, nos encargaremos de ver las etiquetas relacionadas con objetos, parámetros, applets y variables.

ETIQUETAS DE OBJETOS, APPLETS Y VARIABLES.

ETIQUETA HTML	DESCRIPCIÓN	PRINCIPALES ATRIBUTOS
<object>	Esta etiqueta se utiliza para embeber objetos dentro de un documento HTML. Su soporte no es total en los principales navegadores.	**align**, **archive** (URL del archivo), **border**, **classid** (id con la que se setea en Windows Registry o la URL), **codebase** (URL del código para el objeto), **codetype** (tipo MIME), **data** (URL de datos del objeto), **declare** (objeto solo debe ser declarado), **width**, **height**, **hspace** (espacio horizontal), **vspace** (espacio vertical), **name**, **standby** (texto de espera), **type** (MIME), **usemap** (URL del mapa de imagen usado con el objeto), **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.
<param>	Permite definir parámetros para un objeto o un applet.	**name** (nombre del parámetro usado en el código), **type**, **value**, **valuetype**, **id**.
<applet>	Brinda la posibilidad de insertar un applet de Java. Su uso no es aconsejado; se recomienda utilizar en su lugar **<object>**.	**code** (URL del applet), **object** (la referencia de serialización de representación del applet), **align**, **alt**, **archive**, **codebase**, **height**, **hspace**, **name**, **vspace**, **width**, **class**, **id**, **style**, **title**.
<var>	Permite indicar la instancia de una variable.	**id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

Tabla 8. Etiquetas HTML que podemos emplear en el cuerpo del documento.

FLASH EN HTML5

A pesar de la rivalidad entre **HTML5** y **Flash**, es importante que la versión 5 de HTML da soporte a **SWF**. Mediante el elemento **<objetct>**, es posible embeber una película creada en Flash, indicando su ruta como **data**. Además, asignarle diversos atributos a la película de Flash, como ancho y alto, entre otros.

Figura 10. Para utilizar applets, es necesario que el equipo cliente tenga instalada la Máquina Virtual de Java, **`www.java.com/es`**.

Elementos que se retiran del estándar o cambian con HTML5

Una de las tareas principales que encara el desarrollo de HTML5 es la de retirar del estándar todas aquellas características que resultan obsoletas o que con el tiempo han caído en desuso. En la mayoría de los casos, se refiere a etiquetas o atributos que pueden ser reemplazados por otras técnicas más avanzadas y mejor preparadas. Una recomendación muy importante para tener en cuenta es que todo lo que pueda ser resuelto empleando estilos, debe ser realizado mediante CSS. De esta manera, logramos la finalidad de abstraer la estructura del documento y los datos de la representación que nos

proveen los estilos. Entre los elementos que se retiran, encontramos los relacionados con representación, por ejemplo los siguientes: **<basefont>**, **<big>**, ** **, **<font>**, **<s>**, **<strike>**, **<tt>** y **<u>**. También debemos tener en cuenta que se retiran elementos relacionados con la diagramación mediante marcos, ya que estos afectan la usabilidad, la accesibilidad y navegabilidad; ellos son: **<frame>**, **<frameset>** y **<noframes>**.

Además, se retiran: **<dir>** (su función se puede realizar perfectamente con listas del tipo **<ul>**), **<acronym>** (en su lugar se usa **<abbr>**), **<applet>** (se utiliza **<object>** para incluir applets), **<isindex>** (puede reemplazarse por controles de formulario) y **<noscript>** (pierde sentido en HTML5).

Atributos que dejan de ser soportados

En lo relativo a atributos, se producen algunos cambios en lo que se refiere a la posibilidad de ser soportados por algunos elementos:

- **<head>** deja de soportar **profile**.
- **<html>** deja de soportar el atributo **version**.
- **<link>** y **<a>** dejan de soportar **rev** y **charset**. Por su parte, **<a>** deja de soportar **shape** y **coords**.
- Las etiquetas **<img>** e **<iframe>** dejan de soportar **longdesc**.
- **<link>** deja de soportar el atributo **target**.
- **<area>** ya no soporta **nohref**.
- Ya no debe utilizarse el atributo **name** con **<img>**.
- **<meta>** deja de soportar **scheme**.
- **<object>** deja de soportar los atributos **archive**, **classid**, **codebase**, **codetype**, **declare** y **standby**.
- **valuetype** y **type** son atributos que ya no deben usarse con **<param>**.
- **<td>** y **<th>** ya no soportan **axis** y **abbr**. Por su parte, la etiqueta denominada **<td>** deja de soportar **scope**.

¿MAQUETADO CON TABLAS O SIN ELLAS?

Ya hemos leído anteriormente la recomendación que desaconseja el uso de tablas. Para aclarar el tema, lo que no está recomendado es maquetar utilizando tablas, ya que existen otros elementos destinados para esto y que son más efectivos. Las tablas deben utilizarse para lo que han sido creadas: mostrar información que es necesario presentar de manera tabulada.

Recordemos que **align** (atributo soportado por muchos elementos) no es recomendado, ya que su función puede ser efectuada mediante el uso de CSS. Otras recomendaciones son las siguientes:

- Siempre a favor de CSS, para la etiqueta **<body>** no es recomendable usar **alink**, **link**, **text**, **vlink**, **background** y **bgcolor**.
- Para elementos de tablas, no debemos emplear los atributos **bgcolor**, **border**, **cellpadding**, **valign**, **height**, **width**, **nowrap**, **rules**, **frame**, **char** y **charoff**.
- El atributo **clear** no es recomendado para **
. En el caso de listas y menús, no se recomienda emplear los atributos **compact y **type**.
- Para **<iframe>**, ya no hay que utilizar los atributos llamados **frameborder**, **scrolling**, **marginheight** y **marginwidth**.
- Tampoco es recomendable usar **hspace** y **vspace** con **<img>** y **<object>**.
- Para **<hr>**, no se recomiendan los atributos **size** y **noshade**.

Elementos que cambian con HTML5

Otra de las características relevantes para tener en cuenta con HTML5 es que se introducen algunos importantes cambios en ciertos elementos y, también, en algunos atributos.

Si utilizamos **<a>** sin aplicar **href**, estaremos aplicando un enlace al propio sitio. En el caso de **<address>**, se emplea para secciones; la etiqueta **<b>** es para aplicar estilos, pero no tiene importancia (como la tiene ****) para enfatizar. La etiqueta **<menu>** se redefinió para que pueda ser empleada con los nuevos tipos de menú. En cambio, la etiqueta **<small>** se usa para comentarios al margen o impresiones que utilizan tipografías de pequeño tamaño.

En las próximas páginas, vamos a analizar las características que se incorporan en el estándar a partir de esta versión, así apreciaremos el cambio que propone HTML5 para el desarrollo Web.

LOS LOGROS DE TIM BERNERS-LEE

Timothy John Berners-Lee es el nombre de un famoso ingeniero nacido en Londres durante el año 1955. Es el creador de la *World Wide Web* (WWW), se encargó de definir la primera versión del lenguaje HTML, el protocolo HTTP y el sistema de localización de recursos (más conocido como URL). Tim Berners-Lee es también el director del *World Wide Web Consortium* (W3C).

Características que incorpora HTML5

Si bien resulta importante la limpieza que realiza HTML5, y los cambios en algunos de los elementos y atributos conocidos, es aún más destacado todo lo que trae de nuevo esta versión.

Los nuevos elementos, a grandes rasgos, podrían ubicarse en distintos grupos, los cuales mencionamos a continuación: estructura semántica, multimedia, formulario y características avanzadas.

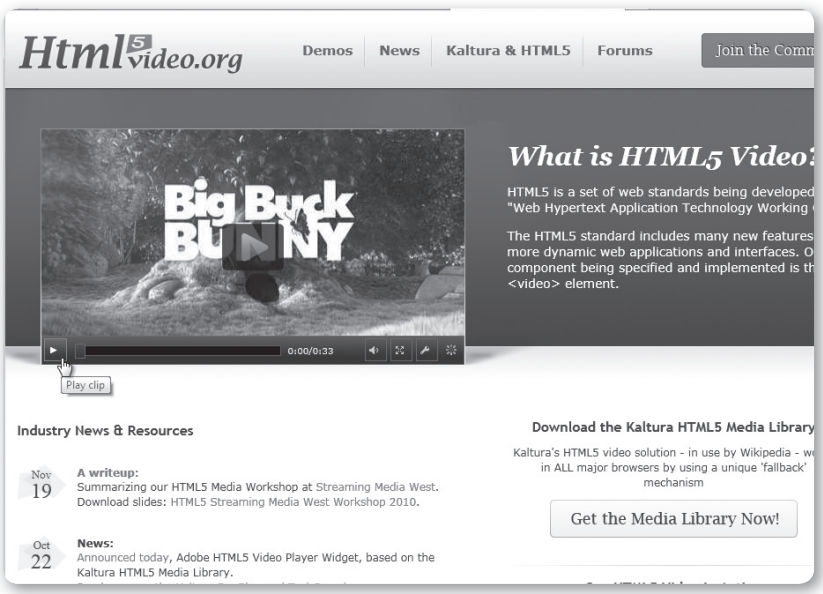


Figura 11. El sitio **`www.html5video.org`** ofrece ejemplos para ver la potencia que pueden alcanzar los videos con HTML5.

Pensando en la estructura semántica encontramos las siguientes etiquetas: **<section>**, **<article>**, **<aside>**, **<hgroup>**, **<header>**, **<footer>**, **<nav>**, **<figure>** y **<figcaption>**. Sobre sus características hablaremos con mayor detalle más adelante, en el **Capítulo 3** de este libro.

Para continuar con los agregados de esta versión, mencionaremos que la etiqueta **<details>** permite describir detalles de un documento

(o de partes de él). Por otro lado, la etiqueta **<summary>** se incorpora para permitir un sumario de los detalles del documento.

También se agregan las etiquetas **<ruby>**, **<rp>** y **<rt>**, que están relacionadas específicamente con anotaciones realizadas con tipografías del Este asiático. Debemos saber que se trata de anotaciones que pueden realizarse debajo de los caracteres en lenguajes como el chino o el japonés, para mostrar su pronunciación o explicación.

Dentro de los elementos que se vinculan con funciones multimedia, se destacan las posibilidades que brindan las nuevas etiquetas **<audio>** y **<video>**, y las opciones de acceso a dispositivo mediante **<device>**, entre otras posibilidades. Veremos más sobre ellos en el **Capítulo 5**.

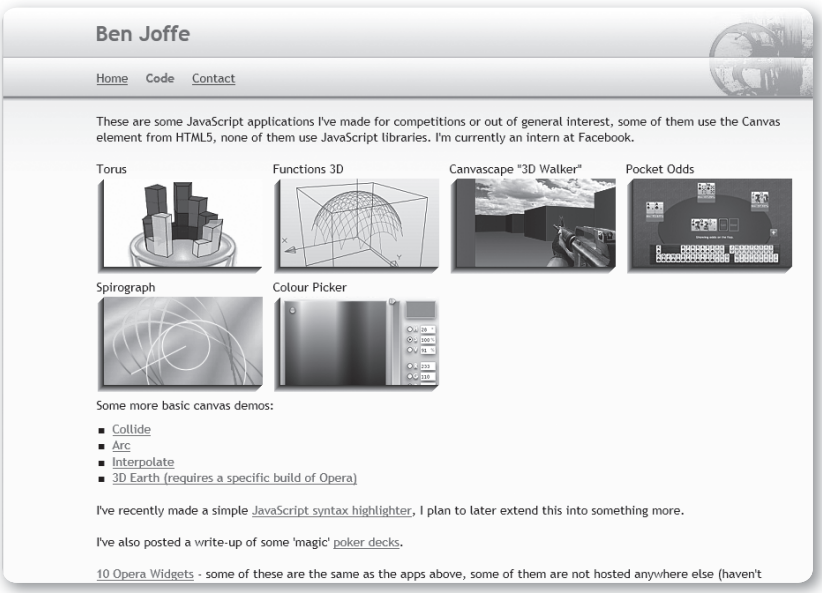


Figura 12. En el sitio **www.benjoffe.com/code**, encontramos muy buenos ejemplos de juegos desarrollados utilizando **<canvas>**.

Para los formularios, los principales cambios llegan en los nuevos tipos (**type**) que soporta el elemento **<input>**; entre ellos encontramos **tel**, **search**, **number**, **range**, **email**, **url**, **datetime**, **datetime-local**, **date**, **month**, **week**, **color** y **time**. Para los elementos de formulario, se incorpora el atributo **autofocus** (enfoque predeterminado cuando carga el formulario)

y, para **<input>** y **<textarea>**, la posibilidad de establecer **placeholder** y **required**. También se agrega la etiqueta llamada **<keygen>** para claves.

Sobre estas características y otras relacionadas con formularios, profundizaremos más adelante en este libro, en el **Capítulo 6**.

Otros elementos que se agregan con HTML5 y cuyas características resultan muy importantes son: **<canvas>** (define un área para contenido dinámico), **<command>** (comandos que pueden ser llamados por el usuario), **<datalist>** (se utiliza junto a un **<imput>** para definir los valores que puede tener), **<details>** (permite describir detalles de un documento o de una parte de él), **<embed>** (para contenidos embebidos, especialmente plugins), **<mark>** (para indicar un texto marcado), **<meter>** (para definir valores de medidas), **<output>** (permite definir un tipo de salida que se utilizará con un script), **<progress>** (para representar, por ejemplo, una barra de progreso), **<time>** (para datos de fechas o temporales) y **<wbr>** (para cortar palabras o líneas).

En HTML5 también se destaca el trabajo que se puede realizar en conjunto con la API de JavaScript. Debemos tener en cuenta que se destaca la posibilidad de realizar Drag & Drop, trabajar con contenidos of ine, geolocalización, almacenamiento local, comunicación bidireccional (con WebSockets) e hilos en paralelo (WebWorker).

Tengamos en cuenta que respecto a estas características avanzadas profundizaremos en el **Capítulo 8** de este libro.

Atributos en HTML5

Entre los principales atributos que se suman a algunos elementos en HTML5, encontramos que **<a>** y **<area>** incorporan **media**. Por otra parte, **<area>** suma los atributos **hre**f **ang** y **rel**. Las listas definidas con **<ol>** suman el atributo **reversed** para indicar que utilizan el orden

TRABAJOS DEL W3C

Para los interesados en los proyectos del World Wide Web Consortium puede ser muy interesante ingresar en el sitio web **www.w3.org/TR/#w3c_all**, verá el listado de los trabajos del W3C que han llegado a ser recomendación junto a su fecha de publicación. En esa misma página, tendremos la posibilidad de conocer los trabajos que están en draft e, incluso, los que han quedado obsoletos.

descendente. Para el caso de la etiqueta **<iframe>**, se incorporan los atributos **sandbox**, **seamless** y **srcdoc**.

Finalmente, haremos referencia a los atributos estándar para los elementos en HTML5. Entre ellos se destacan: **contenteditable** (indica si es editable el contenido), **contextmenu** (define un menú contextual), **draggable** (indica si el elemento se puede arrastrar), **dropzone** (define qué ocurre cuando un contenido es arrastrado), **hidden** (posibilita indicar un elemento que no será mostrado) y **spellcheck** (permite definir si el elemento tiene una definición ortográfica definida).

Es importante saber que los atributos nos permiten definir algunas de las características o capacidades de los elementos HTML. Por esta razón, la incorporación de nuevos atributos potencia aún más las posibilidades con las que cuenta el lenguaje.

Los eventos que se incorporan con HTML5

Debemos tener en cuenta que, en lo que se refiere a eventos, encontraremos muchas características novedosas en HTML5. Entre los nuevos eventos vinculados al objeto denominado **windows** y que se aplican a **<body>** encontramos los siguientes:

- **onafterprint**: ocurre luego de que el documento se imprime.
- **onbeforeprint**: ocurre antes de que el documento sea impreso.
- **onbeforeonload**: se produce antes de que el documento cargue.
- **onerror**: se produce cuando ocurre un error.
- **onhaschange**: se produce cuando el documento cambia.
- **onmessage**: se produce cuando un mensaje es disparado.
- **onoff ine**: ocurre cuando el documento pasa a estado of ine.
- **ononline**: ocurre cuando el documento pasa a estado online.
- **onpagehide**: ocurre cuando la ventana de la página se oculta.
- **onpageshow**: ocurre cuando la ventana se muestra.
- **onpopstate**: ocurre cuando la ventana del historial cambia.
- **onundo**: ocurre cuando se realiza un undo (deshacer).
- **onredo**: ocurre cuando el documento realiza un redo (rehacer).
- **onresize**: se produce cuando la ventana cambia de tamaño.
- **onstorage**: se produce cuando el documento carga.
- **onunload**: ocurre cuando el usuario abandona el documento.

Por otra parte, entre los eventos que se relacionan con el mouse se suman los que mencionamos a continuación:

- **ondrag**: ocurre cuando el elemento es arrastrado (drag).
- **ondragend**: ocurre cuando la operación de arrastrar termina.
- **ondragenter**: cuando el elemento se arrastra a un destino válido.
- **ondragleave**: ocurre cuando el elemento deja un destino válido.
- **ondragover**: este evento ocurre cuando un elemento está siendo arrastrado sobre un elemento válido.

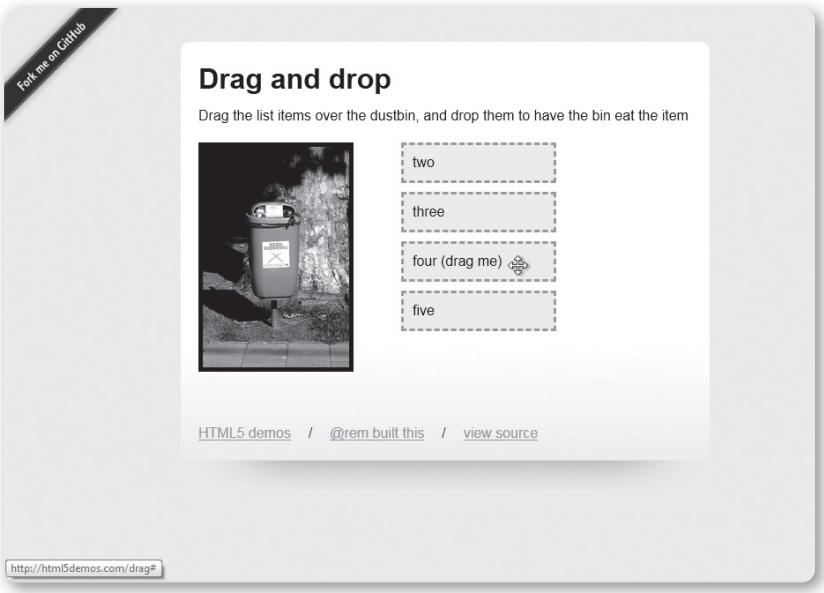


Figura 13. En **`http://html5demos.com/drag`**, veremos un muy buen ejemplo de Drag & Drop (arrastrar y soltar) con HTML5.

ERRORES DE CÓDIGO

Sabemos que al ser un lenguaje interpretado en el navegador, es posible que, en ocasiones, una página HTML se represente aparentemente de manera correcta, a pesar de tener errores de código. No debemos confiarnos de esta situación, ya que esto puede provocar errores en otros navegadores. Evitar errores de código debe ser una regla de oro para nuestros desarrollos.

- **ondragstart**: se produce cuando comienza la operación de arrastre.
- **ondrop**: ocurre cuando el elemento arrastrado es quitado.
- **onmousewheel**: se produce cuando la rueda del mouse es girada.
- **onscroll**: ocurre cuando la barra de scroll del elemento es utilizada.

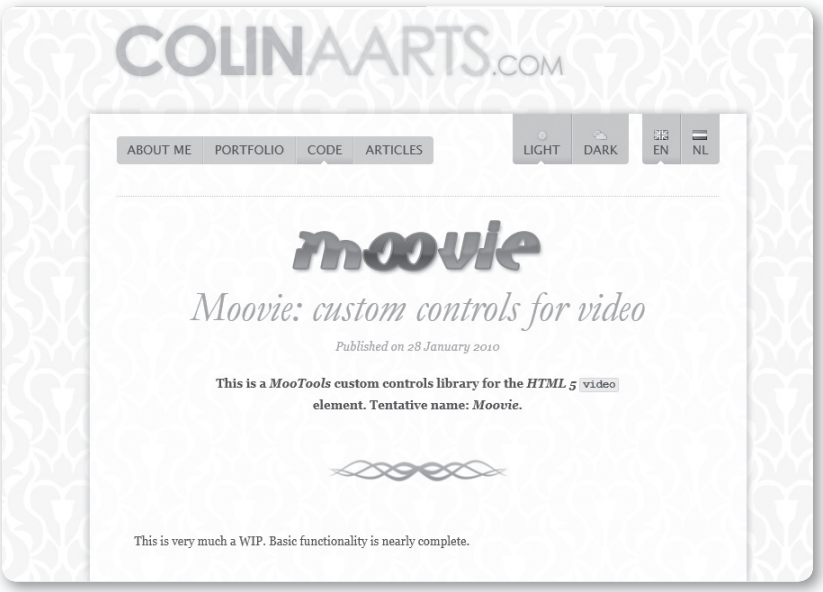


Figura 14. `Moovie` (`http://colinaarts.com/code/moovie`) permite personalizar las características de video provistas por HTML5.

Los eventos relacionados con multimedia (imagen, audio y video) incluyen novedades con:

- **oncanplay**: ocurre cuando el elemento multimedia puede comenzar la reproducción, pero está detenido cargando el buffer.

W3C EN TWITTER

Si utilizamos el popular sistema de Microblogging que nos brinda Twitter (`http://twitter.com`), podremos estar al tanto de las actualizaciones del W3C siguiendo la cuenta **@w3c**. Por su parte, la cuenta de Twitter para seguir las actualizaciones de la Oficina española es **@w3ces**.

- **oncanplaythrough**: ocurre cuando el elemento multimedia puede reproducirse hasta el final (sin detenerse por la carga en el buffer).
- **ondurationchange**: ocurre cuando la duración del elemento cambia.
- **onemptied**: este evento se produce cuando el recuadro multimedia queda vacío de manera repentina.
- **onended**: ocurre cuando el elemento multimedia llega al final.
- **onerror**: este evento se produce cuando ocurre un error durante la carga del elemento multimedia correspondiente.
- **onloadeddata**: ocurre cuando la información multimedia se carga.
- **onloadedmetadata**: ocurre cuando la duración y otra información del elemento multimedia se carga.
- **onloadstart**: ocurre en el momento en el que el navegador comienza a cargar la información multimedia.
- **onpause**: se trata de un evento que se produce cuando se pone en pausa el contenido multimedia.
- **onplay**: este evento ocurre justo cuando el contenido multimedia comienza a realizar la reproducción.
- **onplaying**: se trata de un interesante evento que ocurre cuando el contenido está en proceso de reproducción.
- **onprogress**: se trata de un evento que ocurre cuando el navegador está trayendo el contenido multimedia (en progreso).
- **onratechange**: ocurre cuando la tasa de reproducción cambia.
- **onreadystatechange**: ocurre cuando el ready-state cambia.
- **onseeked**: se produce cuando un elemento multimedia busca un atributo que no es verdadero, y la búsqueda termina.
- **onseeking**: se produce cuando un elemento multimedia busca un atributo que no es verdadero, y la búsqueda comienza.
- **onstalled**: este interesante evento ocurre cuando se presenta un error al traer contenido multimedia (stalled).

¿QUÉ SIGNIFICA INDENTADO?

Indentado es una palabra que se utiliza mucho en el ámbito de desarrollo. Debemos tener en cuenta que no es un término original del idioma español, sino que deriva del inglés **indentation**. Es esencialmente una sangría o tabulado que se coloca entre el margen y el código para mejorar la legibilidad, pero que no tiene ninguna injerencia en su ejecución o en su interpretación.

- **onsuspend**: este evento ocurre cuando el navegador intenta traer contenido multimedia, pero se detiene antes de que todo el archivo o flujo sea recuperado en forma correcta.
- **ontimeupdate**: es un evento relacionado con la actualización en la reproducción, es decir, ocurre cuando el contenido multimedia cambia su posición de reproducciónva.

Figura 15. En **`www.redusers.com/noticias/claves/html5`** podremos encontrar todas las novedades relacionadas con HTML5 publicadas en RedUsers.

ADOBE DREAMWEAVER CS5.5 Y HTML5

La versión CS5.5 de Adobe Dreamweaver da un paso más allá de su predecesor tanto en el soporte de HTML5, como también de estilos CSS3. En esta edición, lanzada en el año 2011, se incorporan mejores características en lo que se refiere a las sugerencias de código para HTML5 y compatibilidad en las vistas. También es posible utilizar layouts preformateadas con macado de HTML5 (de dos y tres columnas) como punto de partida de los diseños en los cuales estemos trabajando.

- **onvolumechange**: es un evento que ocurre cuando se cambia el volumen del elemento multimedia que se está reproduciendo.
- **onwaiting**: se trata de un evento que ocurre cuando se detiene la reproducción, pero se puede esperar a que continúe.

Para formularios también se incorporan nuevos eventos, entre los que encontramos:

- **oncontextmenu**: se produce cuando un menú contextual es disparado.
- **onformchange**: ocurre cuando el formulario cambia.
- **onforminput**: este evento se produce cuando el formulario correspondiente recibe algo de entrada por parte del usuario.
- **oninput**: ocurre cuando un elemento recibe una entrada del usuario.
- **oninvalid**: ocurre cuando el elemento es inválido.

Como hemos podido apreciar en este apartado, son numerosos los cambios que se introducen en lo que se refiere a eventos. Como veremos a lo largo del libro, esto está relacionado con la evolución que impulsa HTML5 para integrarse con diversas tecnologías y nuevas técnicas de desarrollo.

El potencial de HTML5 va mucho más allá de las etiquetas, ya que plantea una evolución importante en la creación de sitios y aplicaciones Web que veremos de aquí a los próximos años.

RESUMEN

En este capítulo, hemos realizado un análisis sobre los elementos que se incluyen en el estándar en HTML4/XHTML. Además, realizamos una revisión sobre los principales atributos y eventos soportados. A continuación, efectuamos una recorrida por las características que se retiran del estándar a partir de HTML5. Luego vimos aquellas características que cambian. Para finalizar, nos detuvimos en los elementos, atributos y eventos que se suman a partir de la quinta versión del lenguaje de marcado de hipertexto.

Actividades

TEST DE AUTOEVALUACIÓN

1	¿Qué son los atributos y para qué se utilizan en el lenguaje HTML?
2	¿Para qué se utiliza la etiqueta **<title>**?
3	¿Para qué se emplea la etiqueta **<body>**?
4	¿Qué diferencias hay al especificar el **DOCTYPE** de un documento HTML 4.01 y uno XHTML?
5	¿Qué elementos se incorporan con HTML5?
6	Indique los elementos que se retiran a partir de la llegada de HTML5.
7	¿Para qué sirve la etiqueta **<canvas>**?
8	¿Cuáles son los atributos que se incorporan con HTML5?
9	De los eventos vinculados con el mouse, ¿cuáles existían en HTML4/XHTML y cuáles se suman con HTML5?
10	¿Qué eventos relacionados con multimedia se incorporan a partir de HTML5?

EJERCICIOS PRÁCTICOS

1	Analice el código de su sitio web y verifique si está utilizando etiquetas no recomendadas por el estándar HTML5. Cambie las etiquetas por las que correspondan.
2	Verifique las etiquetas **<meta>** de su sitio para asegurarse de que cada página tiene una descripción personalizada.
3	Incorpore un archivo SWF en un documento HTML.
4	Instale la Máquina Virtual de Java en un equipo con Windows.
5	Ingrese en **www.html5video.org** y pruebe los contenidos de video HTML5 en diferentes navegadores para verificar el resultado y la compatibilidad.

*03 –

Elementos estructurales y semántica

En este capítulo, vamos a profundizar sobre la importancia que cobra la semántica en el trabajo con HTML5. Analizaremos cómo se debe pensar la estructura de los documentos en HTML5 y vamos a aprender a utilizar, de forma correcta, los elementos destinados a dicho fin. Además, nos introduciremos en temas como Microdatos (Microdata) y también en lo relacionado con SEO (*Search Engine Optimization*).

- **Semántica 100**
- **Declaración de página y cabecera del documento 102**
- **Estructura semántica de documentos en HTML5..... 104**
  - Atributos soportados..... 108
- **Estructurar una página en HTML5..... 110**
  - Adaptar la estructura semántica 111
  - Pensar la estructura de páginas HTML5 desde cero..... 113
  - Compatibilidad..... 119
- **Microdatos..... 120**
- **SEO 122**
- **Resumen..... 125**
- **Actividades..... 126**

Semántica

Sabemos que en nuestra lengua, la semántica se refiere al significado que tienen las estructuras y elementos lingüísticos que la componen; esto se aplica en forma similar a los lenguajes de programación.

Cuando hablamos de un lenguaje de etiquetas, como HTML, la semántica se refiere al significado que tienen los elementos.

Veamos un caso concreto para comprenderlo mejor. Si pensamos en la etiqueta **<h1>**, sabemos que se aplica sobre títulos que tengan la máxima jerarquía en una página web. Su valor predeterminado en la representación de la mayoría de los navegadores está definido por una tipografía Serif (por lo general, Times New Roman) y un tamaño de 2 em. Ahora bien, para la semántica, lo importante de **<h1>** es definir un elemento que será el título principal del contenido de esa página; los aspectos de representación son características que pueden (y deben) ser definidos desde los estilos que corresponden a CSS.

Es necesario recordar que este concepto debemos trasladarlo al resto de los elementos de HTML y tener en cuenta que su uso debe estar relacionado con el objetivo para el cual han sido creados en el lenguaje y no, en la manera en que se representan por defecto.

Como vimos en el capítulo anterior al definir todos los elementos del estándar, hay una finalidad específica para cada etiqueta y, como analizaremos más adelante en el libro, su representación puede ser modificada según el dispositivo de salida.

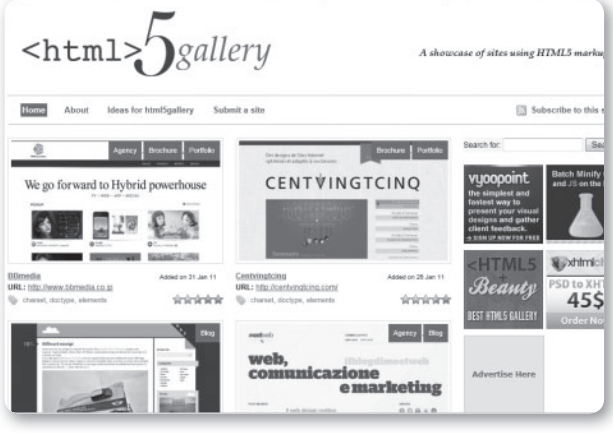


Figura 1. En **http://html5gallery.com** se brinda una interesante galería de sitios web que utilizan HTML5 en su estructura.

La Web semántica siempre estuvo en la cabeza de Tim Berners-Lee desde los comienzos de sus proyectos. Y es ahora, en esta nueva etapa de la web, con la aparición del lenguaje HTML5 como estandarte, cuando el sueño del padre de la Web por fin comienza a tomar un papel mucho más relevante.

Con la web semántica se abren nuevos caminos. Por ejemplo, es posible que los agentes informáticos logren reconocer los elementos por su significado semántico y, posteriormente, puedan guardar esa información para que, en otra etapa, se actúe sobre la base de ella. Esto contribuirá también a la creación de aplicaciones y dispositivos cada vez más robustos e inteligentes, que podrán resolver mayor cantidad de situaciones de manera totalmente automática.

En este capítulo, profundizaremos sobre la semántica de HTML5 y veremos cómo es posible comenzar a construir la estructura de nuestro sitio aprovechando las características mencionadas.

Figura 2. Podemos acceder a templates que usan HTML5 en la siguiente dirección web: **`http://bit.ly/qnEUdU`**.

SEMÁNTICA

Debemos tener en cuenta que el uso de la semántica no es exclusivo del lenguaje HTML, que este concepto se aplica también a los lenguajes de programación en general. Es interesante saber que este término se refiere al significado de la construcción que se realiza en el lenguaje. Por su parte, tengamos presente que la sintaxis se refiere a la gramática propia del lenguaje.

Declaración de página y cabecera del documento

Cuando observamos una página HTML, en primer lugar encontraremos la declaración del tipo de documento. Una de las buenas noticias que nos trae HTML5 es, precisamente, la simplificación de la declaración del tipo de documento. A partir de ahora, solo debemos indicar: **<!DOCTYPE html>**. Es importante recordar que esta declaración se ubica antes de escribir la etiqueta **<head>**.

Dentro de **<head>**, debemos incluir el título **<title>** y también las etiquetas **<meta>** para los metadatos.

En lo que se refiere a la etiqueta **<meta>**, en HTML5 soporta los atributos **charset** (nuevo en HTML5, permite definir la codificación de los caracteres), **content**, **http-equiv** y **name**, deja de soportar **scheme**.

Entre los atributos **<meta>**, es importante resaltar los valores que puede recibir **name** a partir de HTML5 (debemos tener en cuenta que algunos de ellos son heredados de HTML4):

- **author**: autor del documento.
- **copyright**: información de copyright del documento.
- **description**: descripción del documento.
- **distribution**: define el nivel de distribución del documento.
- **generator**: permite especificar con qué programa se creó el documento.
- **keywords**: posibilita introducir las palabras clave relacionadas con el documento (se escriben separadas por comas).
- **progid**: este atributo nos permite indicar la id del programa que estamos usando para generar el documento.
- **rating**: permite indicar el rating de la página.
- **resource-type**: permite indicar el tipo de recurso.

ETIQUETAS Y ELEMENTOS DE HTML

A partir de este capítulo, nos preocuparemos por ir profundizando la información sobre el uso de los elementos presentes en el lenguaje HTML5. Para poder seguir este material, es recomendable conocer la función que cumple cada una de las etiquetas de este lenguaje. En caso de tener dudas sobre este tema, será necesario repasar el contenido que analizamos en el **Capítulo 2**.

- **revisit-after**: permite definir la tasa de actualización de la página.
- **robots**: brinda la posibilidad de indicar reglas para los robots.
- **others**: se puede utilizar para definir **names** propios en el esquema.

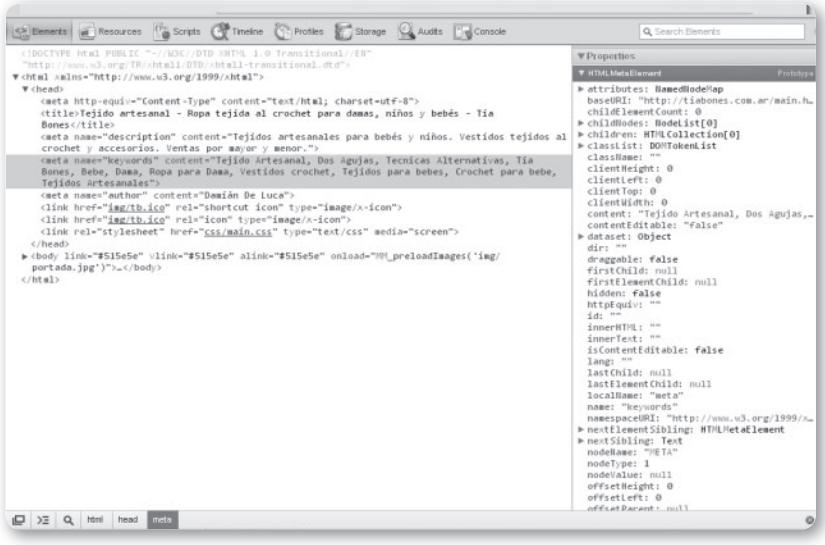


Figura 3. Si realizamos la inspección de una página con el inspector de Google Chrome, entre otros detalles podremos obtener una vista de la cabecera del documento y de sus propiedades.

A continuación, vemos cómo aplicar el título y las principales opciones de la etiqueta de metadatos en el encabezado de un documento. De esta forma estaremos preparados para enfrentar la creación de un documento teniendo en cuenta estos importantes elementos.

MAPA DE SITIO

Debemos tener en cuenta que un mapa de sitio (o sitemap) es una página en formato XML o HTML que presenta los enlaces de las páginas que conforman el sitio web, por lo general, de manera jerárquica. Este elemento es importante para el SEO y el rastreo de los navegadores. También resulta útil en relación con la usabilidad y la accesibilidad de un sitio web.

```
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>RedUSERS</title>
        <meta name="description" content=" Comunidad de Tecnología"  />
        <meta name="keywords" content="Informática,Software,Hardware,
        Gadgets,Internet,Juego,Tutoriales" />
        <meta name="author" content="RedUSERS" />
</head>
```

Estructura semántica de documentos en HTML5

Recordemos que conocer el significado semántico de los elementos que componen los lenguajes que interactúan en la Web, resulta fundamental para crear cimientos sólidos para todas las páginas que conforman la estructura de nuestros sitios Web.

Como ya hemos mencionado, una de las características más valoradas de HTML5 está relacionada con la semántica. En este sentido, HTML5 introduce elementos específicos para poder definir secciones del documento y también características que pretenden hacer de la semántica una capacidad importante para el lenguaje.

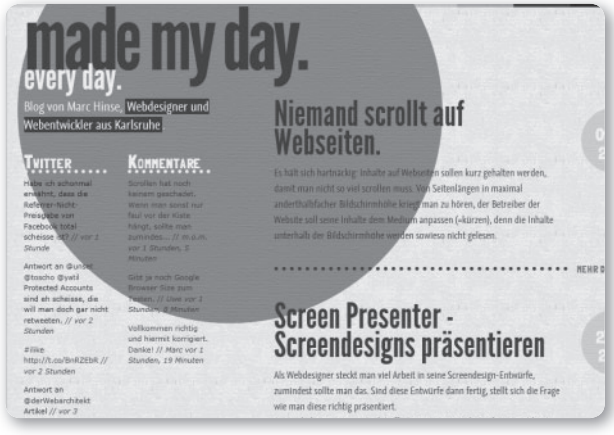


Figura 4. En **`http://www.mademyday.de`** podemos ver una buena combinación de uso semántico de HTML5 y un excelente diseño.

En lo que se refiere a la estructura del documento, en HTML4 estábamos acostumbrados a definir las partes del cuerpo mediante el uso de la etiqueta ** **. El problema se planteaba en la imposibilidad de asignarles la semántica correspondiente a las diferentes partes. Por ejemplo, si bien podíamos aplicar una **id** con el valor **nav** o **footer** (barra de navegación y pie), esto no era más que el valor de un atributo y no le daba un significado semántico diferente al elemento.

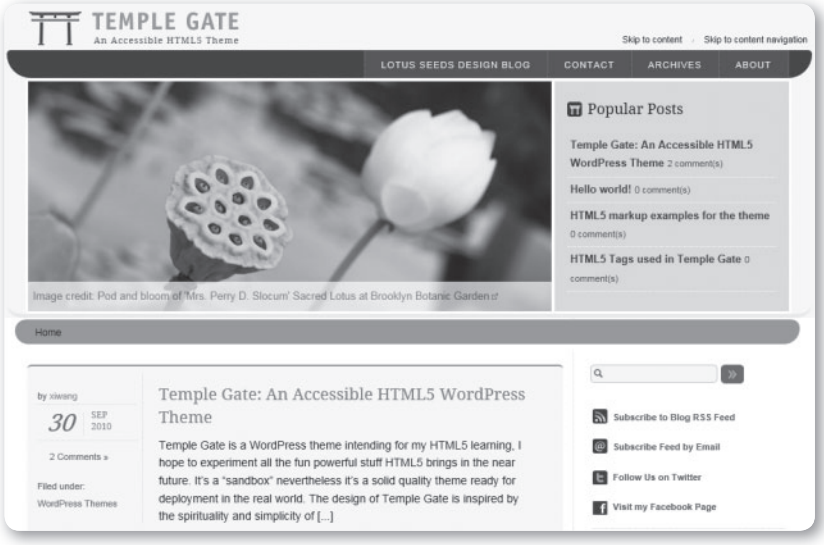


Figura 5. Si realizamos desarrollos con WordPress, podremos encontrar numerosos themes que utilizan semántica HTML5 en la dirección **www.lotusseedsdesign.com/temple-gate**.

A partir de HTML5 se definen etiquetas que nos permiten estructurar el cuerpo de una página con una semántica específica para cada elemento. Entre estas etiquetas encontramos:

- **<header>**: se utiliza para definir la cabecera de la página. No hay que confundir esta etiqueta con **<head>**, ya que **<header>** se emplea para elementos del cuerpo del documento.
- **<hgroup>**: brinda la posibilidad de agrupar títulos con subtítulos. Por ejemplo, en el caso de que, luego de un título **<h1>**, ubiquemos un subtítulo **<h2>** que se relaciona de manera directa con el primero.

- **<nav>**: esta etiqueta está pensada para definir la barra de navegación del sitio web. Aquí podremos definir enlaces internos y también hacia otros sitios. Cabe destacar que no todo conjunto de enlaces es una barra de navegación.
- **<section>**: se encarga de definir una sección de contenido que se representa en la página. Vale destacar que no se trata de un contenedor genérico, como puede ser un ** **.
- **<article>**: sirve para definir artículos o contenidos que pueden ser individualizados. Por ejemplo, se podría utilizar para contener cada noticia o post en un sitio de noticias o blog. En ese contexto, también podría utilizarse para individualizar comentarios.
- **<aside>**: se puede emplear para todo el resto de las cosas que se incluyen en el sitio y no está directamente relacionado con el contenido principal de dicha página (aunque sí puede estar referido de alguna manera). Puede utilizarse como sidebar en un sitio web o blog, aunque esa no es su única aplicación.
- **<footer>**: se utiliza para el pie de la página.

Más adelante, en este mismo capítulo, veremos cómo realizar una estructura utilizando las etiquetas que analizamos.

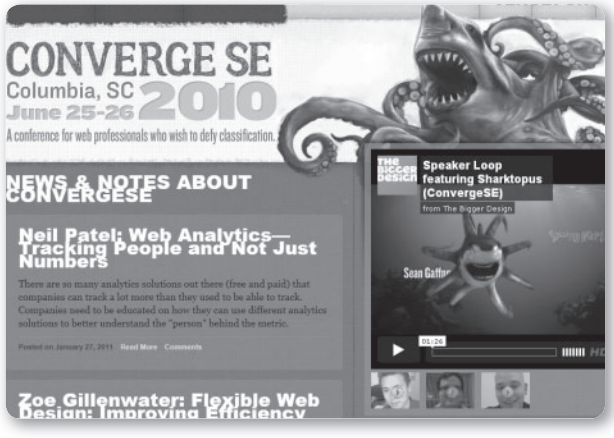


Figura 6. El sitio **`http://convergese.com`** es un muy buen ejemplo de la semántica HTML5 aplicada a la estructura de las páginas.

También encontramos la etiqueta denominada **<figure>**, que puede actuar de manera independiente o junto con **<figcaption>**. A continuación, nos preocuparemos de analizar un ejemplo que agrupa varias imágenes con un epígrafe descriptivo.

```
<figure>
  <img src="./img/iphone.jpg" alt="iPhone de Apple">
  <img src="./img/galaxy.jpg" alt="Samsung Galaxy">
  <img src="./img/xperia.jpg" alt="Xperia de Sony Ericsson">
  <figcaption>Diferentes modelos y marcas de teléfonos móviles inteligentes.</
figcaption>
</figure>
```

Figura 7. Aquí observamos el resultado del ejemplo del uso de **<figure>** y **<figcaption>**, representado en el navegador.

La etiqueta **<time>** nos permite incluir información de hora (en formato de 24 horas) o fecha del calendario Gregoriano (también podemos incluir opcionalmente la hora).

Con la etiqueta **<details>**, es posible describir detalles de un documento (o de algunas partes de este). Con **<summary>**, se puede especificar un sumario que ofrezca detalles del documento.

En lo referido al texto, debemos saber que se incorpora **<mark>** para indicar que un texto está marcado o destacado.

Como podemos ver, las etiquetas que nos permiten definir una estructura semántica en nuestros documentos HTML nos brindan la posibilidad de dar un paso adelante en un código mucho más claro de leer, tanto para los desarrolladores como también para los agentes informáticos, por ejemplo, los robots de los buscadores o los dispositivos electrónicos que requieren estas particularidades para facilitar las características de accesibilidad.

Atributos soportados

Entre los atributos que soportan los elementos **<header>**, **<hgroup>**, **<nav>**, **<section>**, **<article>**, **<footer>**, **<figure>** y **<figcaption>**, encontramos algunos que se heredan de la versión 4 de HTML: **accesskey**, **class**, **dir**, **id**, **lang**, **style**, **tabindex** y **title**.

Para estos elementos, también podemos aplicar los nuevos atributos de HTML5, entre los que figuran: **contenteditable** (soporta **true** o **false** y permite indicar si un elemento puede ser editado o no por el usuario), **contextmenu** (permite especificar el menú contextual para un elemento), **draggable** (soporta **true**, **false** o **auto**; permite indicar si un elemento puede ser arrastrado), **dropzone** (soporta **copy**, **move** o **link**; permite indicar qué hacer cuando un ítem es arrastrado sobre ese elemento), **hidden** (se utiliza para indicar cuando un elemento se encuentra oculto) y **spellcheck** (tengamos en cuenta que se emplea para la ortografía y gramática de un elemento).

LOS ATRIBUTOS PERMITEN AGREGAR INFORMACIÓN O CARACTERÍSTICAS ADICIONALES A LOS ELEMENTOS.

Por parte de **<time>**, entre los atributos encontramos **datetime** (se utiliza si la fecha y la hora no se incluyen en el elemento y deseamos especificarlas por medio de este atributo). El otro atributo soportado es **pubdate** (booleano); cuando se coloca, permite especificar si la hora y fecha del elemento son las que corresponden al documento o al elemento antecesor (**<article>**) que resulta más próximo. Para cada artículo (**<article>**) no debería haber más de un elemento de tiempo (**<time>**) con el atributo **pubdate** definido.

Por otro lado, **<details>** soporta el atributo **open**, que permite indicar si está abierto el detalle, es decir, si es visible; **<summary>** solo soporta los atributos globales de HTML5.

El atributo rel

En las versiones anteriores de HTML/XHTML, conocimos la importancia del atributo **rel** para agregarles información a los enlaces. En el caso de usar este atributo con **<link>**, le brindamos más información al navegador; si lo usamos con **<a>**, le estaremos incluyendo datos que podrán ser útiles para los buscadores. También puede utilizarse con la etiqueta **<area>**.

Básicamente, el atributo **rel** nos permite establecer la relación entre el documento actual y el que se está vinculando.

Entre los valores que puede adquirir este atributo en el lenguaje HTML5 encontramos los que vemos en el siguiente listado:

- **alternate**: vincula a una versión alternativa del documento.
- **archives**: enlace a información histórica.
- **author**: enlace a información relativa al autor del documento.
- **bookmark**: vínculo a dirección permanente del documento.
- **external**: enlace a un documento externo.
- **first**: vínculo al primer documento de una selección.
- **help**: enlace a un documento de ayuda.
- **icon**: vínculo al icono del documento.
- **index**: vínculo al índice del documento.
- **last**: vínculo al último documento de una selección.
- **licence**: vínculo a la información de licencia del documento.
- **next**: vínculo al siguiente enlace de una selección.
- **nofollow**: comúnmente se utiliza para indicar que los robots de los buscadores no sigan el link correspondiente.
- **noreferrer**: se trata de un valor que permite especificar que el navegador no debe enviar **referrer** en las cabeceras HTTP, es decir, que no indique desde qué página se llega a la siguiente.
- **pingback**: la URL para hacer pingback.
- **prefetch**: permite indicar que el documento debe ser cacheado.
- **prev**: vínculo al documento previo de una selección.
- **search**: se encarga de establecer un vínculo a la herramienta de búsqueda del documento correspondiente.
- **sidebar**: se utiliza para el vínculo de los elementos secundarios del documento, los que se ubican en la sidebar.
- **stylesheet**: vínculo a la hoja de estilos que se importa en el documento.

NAVEGABILIDAD

El concepto de **navegabilidad** hace referencia a la facilidad que tiene un visitante para moverse dentro de las secciones y páginas que conforman un sitio web. El diseñador siempre debe pensar una interfaz web clara, con opciones que le ofrezcan al usuario la información necesaria para saber dónde está ubicado, dónde ha estado anteriormente y hacia dónde puede dirigirse.

- **tag**: etiqueta del documento actual.
- **up**: permite indicar que el enlace está por encima del actual en un árbol jerárquico de documentos.

Vale aclarar que, en esta lista, se incluyen tanto los valores heredados que siguen vigentes así como también los que se incorporaron a partir de la versión 5 de HTML.

Los valores **bookmark**, **external**, **nofollow** y **noreferrer** no son soportados por **<link>**. Por su parte, **icon**, **pingback**, **prefetch** y **stylesheet** no pueden utilizarse con **<a>** ni con **<area>**.

Figura 8. Un uso que se le puede dar al valor **`prev`**, **`next`**, **`first`** y **`last`** puede relacionarse con la paginación.

Estructurar una página en HTML5

Con lo que ya hemos visto en este capítulo, tenemos suficientes datos para poder enfrentar la creación de nuestra estructura de página utilizando la semántica de HTML5, aunque aún debemos revisar muchos conceptos importantes.

Ahora bien, el primer dilema que deberemos resolver es si vamos a trabajar con un proyecto existente, migrándolo desde HTML4/XHTML a HTML5, o bien arrancaremos desde cero con el uso de HTML5. A continuación, analizaremos ambas alternativas.

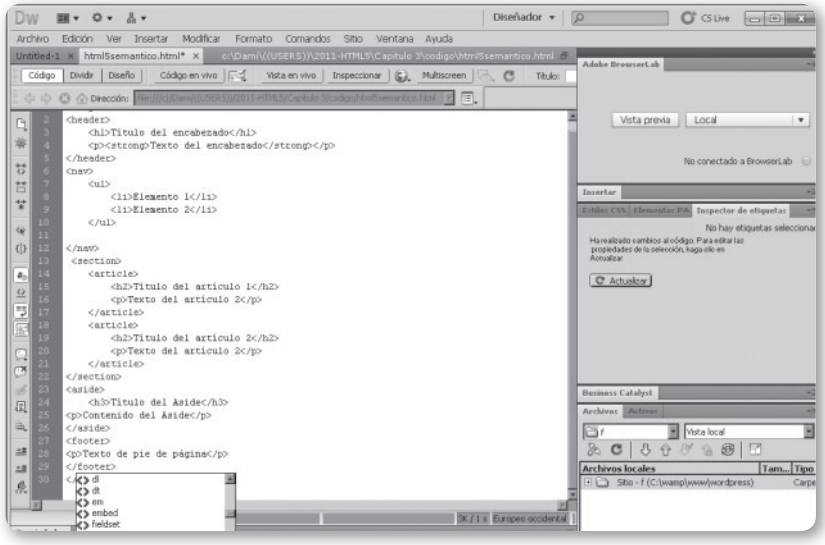


Figura 9. Cuando comenzamos a trabajar con HTML5, es recomendable contar con un editor que nos ayude con las etiquetas de este, como por ejemplo, Dreamweaver CS5 (con la actualización de HTML5 y CSS3).

Adaptar la estructura semántica

Antes de migrar hacia HTML5, debemos plantearnos si el desarrollo del sitio está lo suficientemente sólido en XHTML. Es decir, un maquetado realizado con tablas, errores semánticos, códigos mal estructurados, estilos aplicados inline y etiquetas que no cierran, sin dudas, no resulta un buen punto de partida. Si nuestro desarrollo se encuentra en este estado, es mejor considerar un comienzo desde cero y saltar al siguiente apartado de este capítulo.

SITEMAP.ORG

Si ingresamos en en el sitio web **www.sitemaps.org/es/index.php**, encontraremos información sobre la importancia de la construcción de un mapa de sitio, el modo de uso del formato XML para crear mapas de sitios y ejemplos. También hallaremos información sobre cómo agrupar varios mapas de sitios, y una sección de preguntas frecuentes, entre otros datos muy interesantes.

En caso de contar con un código limpio, creado a conciencia y que respeta las reglas de XHTML, tendremos una base interesante para que podamos realizar la migración a HTML5 de mejor forma y, sobre este patrón, nos basaremos en este caso.

A continuación, veremos un código de ejemplo de estructura en HTML4 simple, basada en etiquetas ** **:

```
<div id="header"> <!--Inicio header -->
        <h1>Header</h1>
</div>            <!--Fin header -->
<div id="nav"> <!--Inicio Nav -->
        <p>Nav</p>
</div> <!--Fin nav -->
<div id="section"> <!--Inicio Section -->
        <p>section</p>
</div> <!--Fin section -->
<div id="aside"> <!--Inicio aside -->
        <p>aside</p>
</div>            <!--Fin aside -->
<div id="footer">  <!--Inicio footer -->
        <p>footer</p>
</div> <!--Fin footer -->
```

En este ejemplo, el lector podrá ver que se utilizaron comentarios HTML (se inician con **<!—**y se cierran con **-->**) para indicar el principio y el final de cada bloque. El uso de estos comentarios puede ser útil cuando se trabaja con códigos extensos.

SITIOS Y CONTENIDOS PARA HUMANOS

Es importante tener en cuenta que, en algunos casos, el deseo de ganar posiciones en los buscadores puede llevar a un exceso de optimización de los sitios web en los cuales estamos trabajando; debemos saber que esta es una situación delicada ya que en algunos casos podría jugarnos en contra. Algo que no debemos perder de vista nunca es que el sitio está orientado a los usuarios que lo visitarán, por eso debe ser agradable para que ellos lo recorran y que su contenido les resulte útil. De esa manera, si han encontrado lo que buscaban al ingresar, seguramente volverán a visitarlo.

En el código que presentamos a continuación, veremos la estructura adaptada a la semántica propia de HTML5:

```
<header>
        <hl>Header</hl>
</header>
 <nav>
        <p>Nav</p>
</nav>
<section>
        <p>section</p>
</section>
<aside>
        <p>aside</p>
</aside>
<footer>
        <p>footer</p>
</footer>
```

En nuestro ejemplo, observamos que la semántica de HTML5 aporta mucha claridad y no se requiere la presencia de comentarios para un caso como este. No obstante, los comentarios que se utilizaban en HTML siguen vigentes en la quinta versión de este estándar y son recomendables para códigos que lo ameriten, en especial en sitios que cuentan con muchas páginas y en los que se trabaja en equipo.

Pensar la estructura de páginas HTML5 desde cero

Si nos encontramos frente a un nuevo proyecto o si hemos decidido rehacer un sitio desde cero, porque estructuralmente no nos ofrecía una buena base, entonces puede ser un buen momento para comenzar a implementar lo que hemos aprendido sobre HTML5.

Tengamos en cuenta que una vez que realizamos las maquetas del sitio, analizamos las necesidades, y verificamos aspectos de usabilidad, accesibilidad y navegabilidad, ya tendremos todo listo para comenzar a andar el camino de estructurarlo con HTML5.

Una recomendación que se hace en estos casos es realizar primero la estructura HTML de las secciones más importantes del cuerpo del documento (cabecera, contenido principal, barra de navegación, pie, etcétera) y, luego, ir agregando los elementos que se incorporan en ellas.

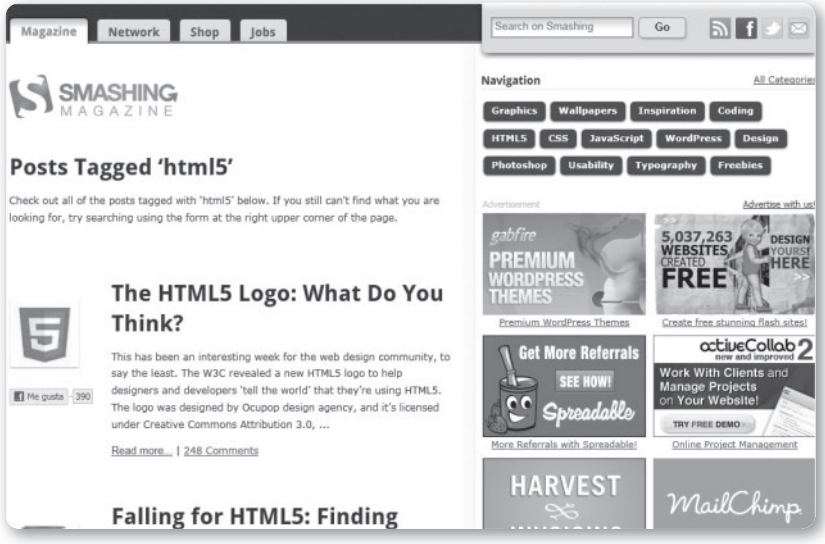


Figura 10. `www.smashingmagazine.com` es un sitio que nos ofrece tutoriales y artículos sobre diseño web. Cuenta con secciones sobre HTML5, CSS, usabilidad y mucho material para inspirar al lector.

A continuación, veremos el ejemplo de un código básico de estructura semántica de HTML5. En este caso, trabajaremos el código del cuerpo del documento con un encabezado **<header>**, una barra

¿QUÉ ES RSS?

Really Simple Syndication (conocido por sus siglas **RSS**) es una de las maneras más populares de compartir y distribuir contenidos en Internet, aprovechando las ventajas del formato XML. Si tenemos un sitio de noticias, un blog o cualquier web de actualización frecuente, se recomienda que incorporemos esta característica para lograr redifusión de los contenidos.

de navegación **<nav>**, seguido de un bloque principal denominado **<section>** que contiene dos artículos **<article>**, una barra con información adicional **<aside>** y también un pie **<footer>**:

```
<header>

        <h1>Título del encabezado</h1>
    <p><strong>Texto del encabezado</strong></p>
</header>
<nav>

        <ul>
                <li>Elemento 1</li>
                <li>Elemento 2</li>
        </ul>

</nav>
 <section>

        <article>
                <h2>Título del artículo 1</h2>
                <p>Texto del artículo 2</p>
        </article>
        <article>
                <h2>Título del artículo 2</h2>
                <p>Texto del artículo 2</p>
        </article>

</section>
<aside>
        <h3>Título del Aside</h3>
        <p>Contenido del Aside</p>
</aside>
<footer>
        <p>Texto de pie de página</p>

</footer>
```

Figura 11. Aquí vemos la representación del código de estructura HTML5 con los estilos CSS aplicados, sobre los que hablaremos más adelante.

Ahora que hemos visto el modelo, vamos a pensar cómo lo llevaríamos a cabo en la vida real, por ejemplo, en un sitio de noticias o en un blog. Antes de continuar, vale aclarar que esta es simplemente la estructura HTML, y que todo lo relacionado con la apariencia y modo de representación de los elementos en pantalla, lo manejaremos a través de CSS, cuyas reglas veremos en detalle en el capítulo siguiente.

Volviendo a nuestro ejemplo, vamos a comenzar con el **<header>**. En este caso, aplicamos la etiqueta **<h1>** al título del sitio o blog, que

LENGUAJE UNIFICADO DE MODELADO (UML)

Se conoce bajo el nombre de **Lenguaje Unificado de Modelado** (en inglés, Unified Modeling Language o sus siglas **UML**) al estándar que permite describir procesos y métodos de funcionamiento de software. Este lenguaje se utiliza tanto para la creación como para la documentación de aplicaciones, y nos facilita la planificación, resolución y el análisis de proyectos de gran envergadura.

aparece en su página principal, ya que es el elemento de texto más importante. Debajo, utilizamos la etiqueta **<p>** y la destacamos con **** para el texto de descripción que suelen tener los sitios, por ejemplo: "Novedades sobre informática general". Cabe destacar que, en algunos sitios, se utilizan imágenes en el encabezado; en esos casos, por una cuestión de accesibilidad y también de SEO, es recomendable utilizar en la etiqueta **<img>** los atributos **alt** y **title** para describir el texto relativo a dicha imagen. Otra alternativa para resolver el encabezado, dependiendo de la necesidad de nuestra estructura, consiste en el uso de la siguiente semántica:

```
<header>
    <hgroup>
        <h1>Título del encabezado</h1>
        <h2>Subtítulo del encabezado</h2>
    </hgroup>
</header>
```

Ahora pasamos a la barra de navegación **<nav>**, que bien podría estar a la izquierda de la sección de contenido o por encima de ella.

En general, si creamos un menú, podremos estructurarlo con una lista, como en este caso, en la que los ítems están constituidos por la enumeración de los elementos. Luego, por medio de CSS, le daremos el estilo que deseemos.

En este ejemplo, **<section>** contiene el resumen de los artículos. En líneas generales, los periódicos online y los blogs ofrecen, en la portada, el titular de las noticias, un breve texto y el enlace para ver el resto de los artículos que corresponden.

En esta estructura simplificada, vemos dos **<article>**; cada uno de ellos cuenta con un título **<h2>** y un texto en un párrafo **<p>** para la breve descripción de la noticia. En estos casos, también se suele añadir una imagen (no olvidemos agregarle los atributos **alt** y **title**) y, también, un enlace (muchas veces aplicado de manera directa al título **<h2>**) para derivar a la noticia completa.

En este caso, el **<aside>** ha sido simplificado en un título **<h3>** y un texto en un **<p>**, pero aquí pueden existir muchas variantes, desde

> EL PLANEAMIENTO DE LA ESTRUCTURA ES FUNDAMENTAL ANTES DE PONERSE A ESCRIBIR EL CÓDIGO.

widgets de AJAX hasta imágenes con vínculos (publicidades, banners, enlaces a otros sitios, etcétera), entre otras numerosas posibilidades.

Tengamos en cuenta que el pie de página, el cual se encuentra definido con el **<footer>**, puede tener un texto sobre el autor, copyright, enlaces internos o hacia sitios amigos, etcétera.

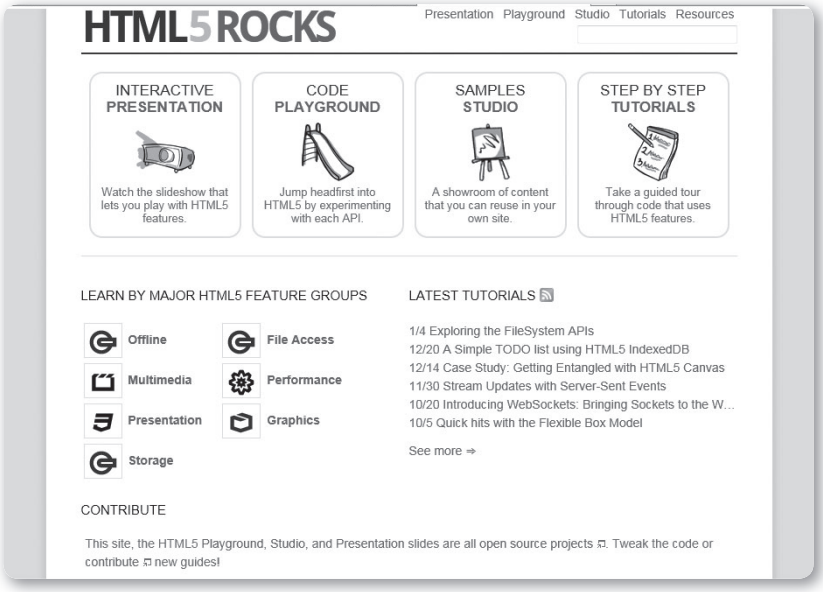


Figura 12. El sitio **www.html5rocks.com** cuenta con una estructura semántica que se basa en este estándar.

La estructura que hemos visto aquí puede adaptarse, fácilmente, a cualquier tipo de necesidad, ya sea un e-commerce, una red social, un foro o cualquier otra opción que podamos imaginar. Luego será solo cuestión de incorporar los elementos para personalizar nuestro proyecto y, con ellos, terminar de darle la estructura deseada a cada página.

Un aspecto importante para tener en cuenta, además del correcto uso semántico de las etiquetas, es el aprovechamiento eficiente de estas. Además de emplear **id** (para identificar de manera única a un elemento) y **class** (para definir clases de elementos), resulta de suma importancia que repasemos los atributos vistos en este capítulo y en el anterior para aplicarlos en cada uno de nuestros proyectos.

Figura 13. Si utilizamos **`Drupal`**, se recomienda visitar **`http://groups.drupal.org/html5`**.

Compatibilidad

Un problema que vamos a enfrentar al pasar a una nueva versión del estándar es el de la **compatibilidad**. ¿Qué ocurrirá con los navegadores que no reconozcan las etiquetas HTML5? En general, no las mostrarán. Esto, sin dudas, se convierte en un problema porque la representación del documento se verá afectada en gran medida y los resultados pueden ser desastrosos. Pero debemos tener en cuenta que la buena noticia es que la migración hacia nuevas versiones de los navegadores trae también una mayor compatibilidad, por esta razón el uso de HTML5 no presentará dificultades.

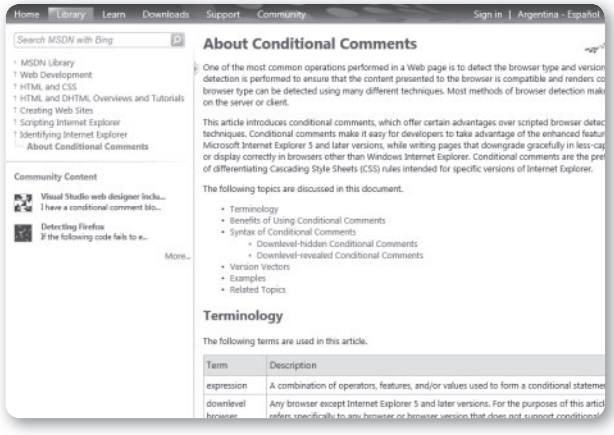


Figura 20. En el sitio de MSDN vemos que los comentarios condicionales sirven para versiones de IE que no soportan HTML5.

Existen ciertos frameworks de AJAX que nos ayudan con el paso a HTML5. En el **Capítulo 1**, hablamos sobre Modernizr, una opción que nos permite detectar las características del navegador del usuario y, de esta manera, poder actuar basándonos en esos datos.

Otra alternativa interesante es **http://52framework.com**. Este framework está enfocado principalmente en el maquetado de sitios web que utilizan HTML5 y CSS3. La gran ventaja reside en que funciona no solo en navegadores modernos, sino que también puede hacerlo en versiones previas, incluso para la familia de Internet Explorer, anteriores a la versión 9 (incluidas las versiones 6, 7 y 8).

Este framework se apoya en JavaScript y CSS para actuar en el maquetado junto a nuestra estructura de documento HTML. Vale destacar que, para quienes están acostumbrados a trabajar con Grids, esta herramienta será especialmente útil para el maquetado.

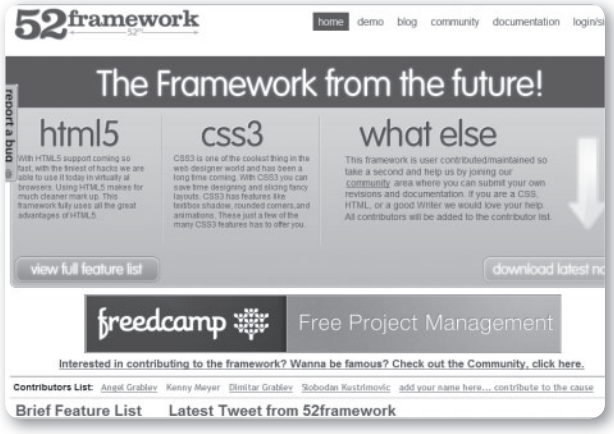


Figura 15. En el sitio web de **`52framework`** es posible acceder a la descarga de esta herramienta y también ver una demo.

Microdatos

Desde hace tiempo, se conocen los microformatos (microformat) en los lenguajes de desarrollo web. En el lenguaje XHTML, utilizando esta característica, era posible incorporar un significado semántico al contenido, con los mismos atributos, tales como **class**, **rel** o **rev**, por ejemplo. Con la llegada de HTML5, se incorporan los **microdatos**

(**microdata**). Es importante entender que la importancia de los microdatos está vinculada con el marcado semántico y la posibilidad de hacerlo más amigable para los agentes informáticos con los que actúe. Por ejemplo, esto puede favorecer al buscador, primero en el rastreo de sus robots y, luego, en el resultado que se ofrece.

Si deseamos proveer más información para los buscadores, en particular al famoso Google, podemos acceder a utilizar lo que se conoce como **Rich Snippets**. Estos son fragmentos que utiliza el buscador para destacar información en los resultados. Esto también le facilita el acceso y le ofrece la información de un solo vistazo al usuario que desea buscar el contenido.

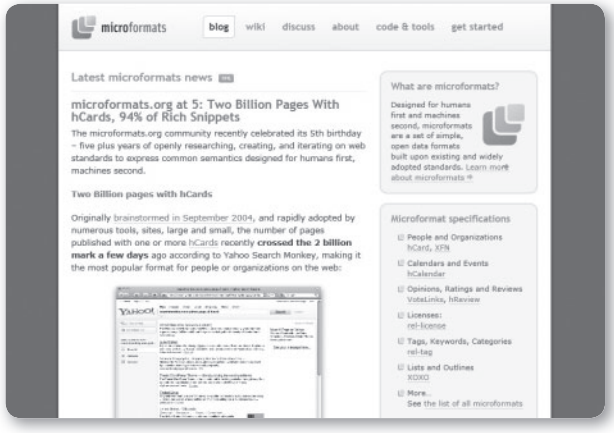


Figura 16. Si buscamos información sobre microformatos, en **`http://microformats.org`** veremos un sitio muy interesante.

Los atributos de los elementos que incorporan los microdatos son: **itemscope** (booleano, indica que se crea un nuevo ítem con pares de valores), **itemtype** (es una dirección URL con una especificación que se pueda aplicar, un diccionario que sirva como vocabulario), **itemid** (es una URL con información relativa a un elemento), **itemref** (permite referenciar a varios elementos mediante la id) e **itemprop** (se utiliza para indicar la propiedad y el valor que corresponde).

Podemos ampliar la información sobre este tema en el sitio web que se encuentra en la dirección **http://dev.w3.org/html5/md**. Vale destacar que esta página del Dev de W3C está en constante actualización, por lo cual encontraremos las novedades más frescas sobre este tema; entonces, es necesario visitarla constantemente.

SEO

El posicionamiento de contenidos de un sitio en los buscadores tiene una larga historia que puede remontarse a la década del noventa.

El término *Search Engine Optimization* (**SEO**) comenzó a utilizarse en el año 1997, y su importancia fue creciendo con el tiempo.

Crear contenidos de calidad, conseguir enlaces entrantes de peso, utilizar títulos de artículos ricos en palabras clave y emplear de manera correcta las metaetiquetas (en especial las de título y descripción) son algunos de los "secretos" conocidos del SEO.

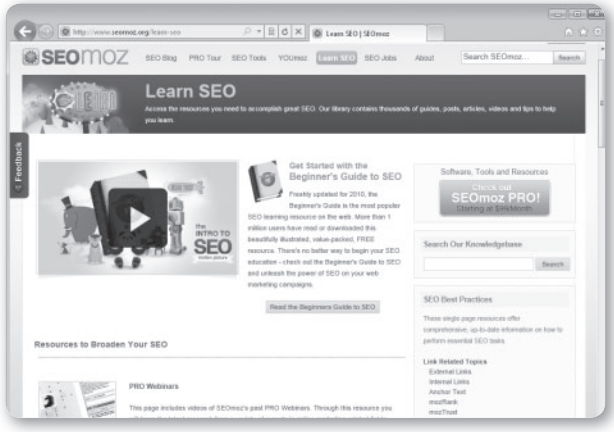


Figura 17. En **`www.seomoz.org`**, encontramos información y videos muy útiles si nos interesa profundizar aún más sobre el tema SEO.

Hoy en día, para lograr posicionar un sitio o una marca en Internet, el SEO debe complementarse con otras técnicas tales como **SEM** (*Search Engine Marketing*) y **SMO** (*Social Media Optimization*), evidentemente más ligadas al marketing y al *Social Media*.

¿QUÉ ES EL PAGERANK?

Para los que están en el mundo del SEO, seguramente el término **PageRank** (**PR**) les sea familiar. Para los que no lo conocen, podemos decir que el PR es un valor o escala numérica, que va desde el 0 al 10, y que es asignado por un algoritmo de Google para determinar la relevancia de una página web. Obviamente, cuanto mayor sea este número, más relevante será la página.

El SEO y sus actividades vinculadas constituyen un material que se sostiene en constante movimiento y en el cual siempre se presentan novedades. Es aquí donde entran a jugar las mejoras que introduce HTML5 en lo que se refiere a semántica.

Figura 18. `Google Instant` propuso un cambio importante en la era de los buscadores y el SEO (`www.google.com/instant`).

¿Por qué razón resultan importantes estos cambios? Porque la posibilidad de estructurar un documento con etiquetas semánticas les dan la oportunidad a los buscadores (y en especial a sus **bots**, los robots de rastreo) de poder saber qué es cada parte del documento. Anteriormente, cuando todo era **`<div>`** o divisiones sin un valor semántico, el buscador no podía asignar un peso específico a cada contenido. Eso cambia ahora, ya que, por ejemplo, se puede saber que lo que está dentro de **`<article>`** es más importante que lo que está en **`<aside>`**.

IMPORTANCIA DE LOS ENLACES

Es necesario tener en cuenta que la tarea de lograr que otros sitios nos enlacen no solo es importante para recibir visitas desde ellos, sino también para favorecer el SEO. Por otro lado, si bien figurar en directorios puede servir, esto ahora tiene mucho menos peso que hace tiempo, cuando se trataba de una de las variables más importantes al momento de posicionar nuestro sitio web. Por estas razones, debemos saber que lo que se recomienda es dedicarnos a la tarea de conseguir enlaces "naturales", es decir, que otros sitios nos enlacen teniendo en cuenta solo la calidad que tiene el nuestro.

Figura 19. `Google Analytics` (`www.google.com/intl/es/analytics`) permite ver los hábitos de navegación de los visitantes.

A continuación veremos una serie de recomendaciones, relacionadas con este tema, que favorecerán nuestros proyectos en vistas a poder un mejor entendimiento con los buscadores:

- Realizar la estructuración de las páginas de nuestro sitio con las etiquetas semánticas que hemos visto en este capítulo: **<header>**, **<hgroup>**, **<nav>**, **<section>**, **<article>**, **<aside>** y **<footer>**.
- Es importante utilizar con criterio las etiquetas de **<h1>** a **<h6>**. Solo utilizar una **<h1>** por página y no abusar de **<h2>** y **<h3>** (ni del resto).
- Evaluar en qué casos es recomendable implementar **<hgroup>**.
- Usar correctamente los atributos **rel** para definir los enlaces del sitio.
- En lo que se refiere a multimedia, es recomendable utilizar los atributos **alt** y **title** para las imágenes. También es importante saber que las etiquetas **<audio>** y **<video>** soportan el atributo **title**.
- Debemos recordar siempre que crear un mapa de sitio contribuirá para una mejor indexación de todos los contenidos.

La construcción de los enlaces internos es muy importante para la navegabilidad del sitio y también para el SEO. Debemos saber que el

anchor text, el texto que se muestra del enlace, es un buen lugar para ubicar las palabras clave de la página interna que se está enlazando.

No olvidemos configurar de manera correcta, mediante **.htaccess**, los contenidos privados del sitio que no deben ser indexados. También es importante revisar y establecer una configuración adecuada para evitar contenidos que puedan ser interpretados y duplicados por los spiders. Si utilizamos un servidor **Apache**, el tutorial que se encuentra en **http://httpd.apache.org/docs/2.2/howto/htaccess.html#how** nos ayudará con esta configuración.

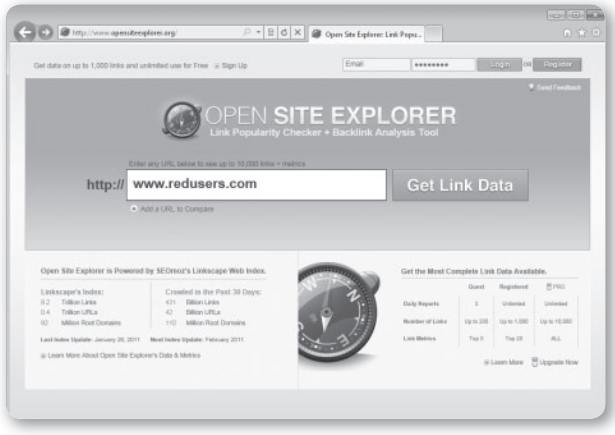


Figura 20. Para conocer datos de autoridad de página y dominio, podemos usar Opensite Explorer (**`http://bit.ly/4DJvSL`**).

RESUMEN

En este capítulo, nos hemos ocupado de hablar sobre semántica, conocimos su importancia en esta nueva etapa de la Web y aprendimos cómo estructurar un documento, empleando las características que incorpora HTML5 para ello. Analizamos las etiquetas que nos permiten dar estructura en HTML5 y observamos cómo llevarlas a la práctica en ejemplos. Conocimos para qué sirven los Microdatos y la importancia del SEO (*Search Engine Optimization*), junto a algunos tips interesantes que nos ayudarán a llevarlo a la práctica en nuestros proyectos web.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué importancia tiene la semántica en la creación de sitios web?
2. ¿Para qué se utiliza la etiqueta **<¡DOCTYPE>**?
3. ¿En qué caso se emplea la etiqueta **<header>**? ¿Es igual que **<head>**?
4. ¿Qué uso debe darse a la etiqueta **<nav>**?
5. ¿A qué parte del documento se aplica la etiqueta **<footer>**?
6. ¿Qué uso se le puede dar a la etiqueta **<aside>**?
7. ¿Qué es Microdata?
8. Indique los diferentes valores de **name** que puede tomar el atributo en HML5.
9. ¿Qué significa SEO?
10. Indique algunas recomendaciones para mejorar el SEO de una página web.
11. actividades prácticas
12. ¿Con cuántos datos se puede llevar a cabo una comparación?

EJERCICIOS PRÁCTICOS

1. Escriba el **DOCTYPE** para un documento HTML5.
2. Escriba las etiquetas **meta** para descripción, palabras clave y autor de una página.
3. Asigne los atributos **rel** correspondientes a un enlace que no desea que sea seguido por las arañas de los buscadores.
4. Adapte la estructura semántica de una página creada en HTML4 o XHTML a las nuevas características de HTML5.
5. Estructure, desde cero, las partes de una página utilizando la semántica de HTML5.

*04

Trabajo con estilos

En los capítulos anteriores, hemos conocido las etiquetas de HTML5 y cómo estructurar nuestras páginas empleando este estándar. Aquí veremos cómo interactúa HTML5 con las hojas de estilo CSS 2.1 y CSS3 para representar los contenidos en el medio de salida. Conoceremos los selectores, clases y propiedades. Recorreremos las propiedades principales de CSS 2.1 y CSS3, y veremos los valores que le podremos aplicar.

- **Interacción de HTML5 con los estilos CSS** **128**
- **Selectores, clases, y propiedades** **129**
- **Construcción de reglas en CSS** **131**
- **Herencia, cascada y especificidad** **134**
- **Estilos con CSS 2.1** **137**
- **CSS3: Nuevas características** **155**
  - Sombra en el texto 156
  - Fondos 167
  - Opciones de animación 182
- **Resumen** **183**
- **Actividades** **184**

Interacción de HTML5 con los estilos CSS

Una de las misiones de HTML5 es afianzar el concepto de que la representación debe ser independiente de la estructura semántica y de los datos del documento con el que se esté trabajando.

Para lograr esto, se retiran del estándar HTML aquellos elementos que se encuentran relacionados con la representación, para que esta necesidad sea resuelta de una manera más eficiente empleando hojas de estilo.

Figura 1. Para estar al día con lo que ocurre con los nuevos estándares, el sitio web **`http://css3html5.com.ar`** nos brinda contenidos, información y novedades sobre CSS3 y HTML5.

En el **Capítulo 1**, hemos hablado sobre la importancia de CSS, conocimos su historia y aprendimos cómo se incorporan los estilos. En este capítulo, profundizaremos sobre el tema, explicando cómo utilizar selectores, identificados, clases, propiedades y atributos. Veremos cómo se construyen las reglas en CSS, qué significa la cascada y qué rol tiene la especificidad cuando hay colisión entre las reglas.

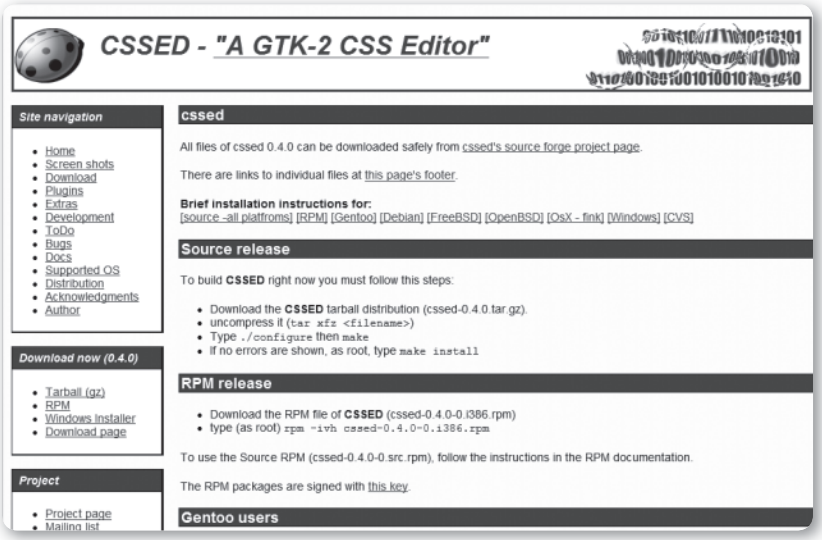


Figura 2. CSSED (`http://cssed.sourceforge.net/download.php`) es Open Source y multiplataforma, nos permite editar y crear código CSS; además, nos brinda soporte para HTML.

Selectores, clases, y propiedades

Antes de comenzar a crear estilos con CSS, es necesario que conozcamos algunos conceptos que nos servirán como base.

Los **selectores** conforman una parte de la regla de estilo, en la que se puede especificar a qué elemento o elementos le aplicaremos el estilo; por ejemplo, el nombre del elemento, su clase o su identificador.

Para poder seleccionar elementos que se encuentran dentro de otros, se emplea lo que se conoce como **selector descendiente**.

El uso de **clases** en CSS permite abstraerse del elemento en particular y construir reglas que se apliquen a toda aquella etiqueta HTML que cuente con el atributo **class** correspondiente. Aquí es importante resaltar que las clases pueden aplicarse a diversos elementos.

Otra característica que puede ser empleada para seleccionar es el **atributo** del elemento HTML, del cual hablaremos más adelante cuando analicemos el tema **Selectores avanzados**.

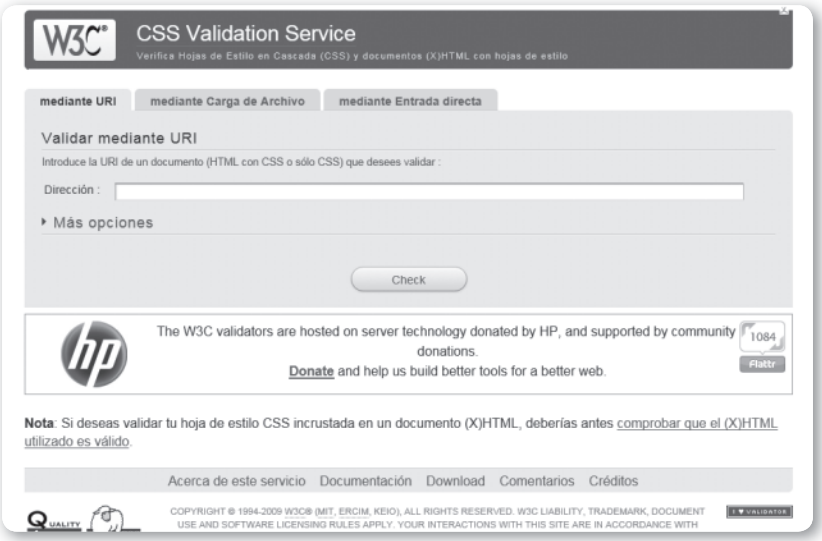


Figura 3. `http://jigsaw.w3.org/css-validator` ofrece un validador de código CSS, que nos brinda un muy buen informe sobre los problemas que pueda encontrar en los documentos.

En la declaración de una regla CSS, los selectores reciben su estilo mediante **propiedades**. Para lograr esto, las propiedades que se declaran en la regla reciben **valores** que determinan la representación del elemento en el medio o medios de salida definidos.

CSS WORKING GROUP

Si ingresamos en el sitio web **www.w3.org/Style/CSS**, encontraremos las últimas noticias sobre CSS. Allí también podremos saber qué está haciendo el grupo de trabajo (CSS Working Group) y acceder a su blog: **www.w3.org/blog/CSS**. Para conocer a sus miembros, podemos ingresar en el sitio web que se encuentra en la dirección **www.w3.org/Style/CSS/members**.

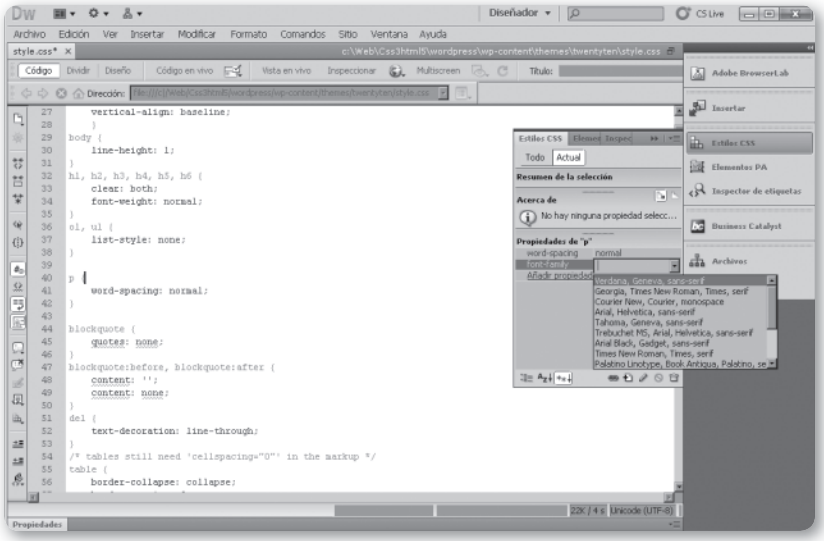


Figura 4. Muchos editores nos ofrecen la posibilidad de detectar la sintaxis de CSS. Con **Dreamweaver** podremos escribir el código o valernos de las cajas de herramientas que ofrece.

Construcción de reglas en CSS

Como sabemos, la sintaxis básica para construir reglas utilizando CSS indica que debemos escribir el selector, abrir llaves, indicar la propiedad y posteriormente asignarle el valor correspondiente. Cerramos la línea con punto y coma, y seguimos agregando pares de propiedad/valor hasta que completemos la declaración (cada una debe estar finalizada con punto y coma). Cuando terminamos, cerramos la llave. La declaración sería entonces:

```
selector {propiedad:valor;}
```

Veamos un ejemplo de esto, aplicando estilo a un párrafo:

```
p {
        font-size: 10px;
        color: blue;
        text-align: right;
    }
```

Ahora crearemos una regla para que los textos destacados con **strong** en el párrafo se muestren en color rojo; para ellos utilizaremos el selector descendiente.

```
p strong {color: red;}
```

LAS REGLAS CSS NOS PERMITEN DEFINIR LA REPRESENTACIÓN DEL DOCUMENTO EN EL NAVEGADOR.

De esta manera, si aplicamos ambas reglas en una hoja de estilos, tendremos el texto del párrafo en color azul y lo que se envuelva dentro del párrafo con la etiqueta **** de color rojo.

Si deseamos aplicar una misma regla a diversos elementos, podemos declararlos antes de abrir la llave, y separarlos con comas; por ejemplo, vamos a aplicarles color gris a todos los títulos de una página.

```
h1, h2, h3, h4, h5, h6 {color:grey;}
```

Cuando construimos reglas para clases, debemos incluir el **.** (punto) antes del nombre, por ejemplo:

```
.nombre_clase {propiedad:valor;}
```

A continuación, veremos un ejemplo donde deseamos asignar color negro al **** cuya clase se llama **marcado**:

```
span.marcado {color:black;}
```

Si creamos reglas para una determinada **id**, debemos anteponerle el símbolo **#**, por ejemplo, **#nombre_id {propiedad:valor;}**.

Para poder aplicar una regla a un elemento que tiene un identificador, debemos indicar el nombre del elemento, luego colocar **#**

(numeral), seguido del nombre del identificador. Encontramos un buen ejemplo: si deseamos poner color de fondo azul a un elemento cuya **id** es **principal** podríamos construir la regla:

```
#principal {background-color:blue;}
```

Si deseamos ser más específicos, a la regla anterior también podríamos indicarle el nombre del elemento que tiene aplicada la **id**.

Otro aspecto que debemos considerar al crear hojas de estilos extensas es que las reglas soportan comentarios, que se inician con **/*** y se cierran con ***/**. Veamos un ejemplo a continuación:

```
/* Este es un comentario de una regla CSS*/
```

Propiedades shorthand

Una de las facilidades que nos ofrece CSS es el uso de métodos abreviados para asignar valores a las propiedades. Esta característica es conocida como **propiedades shorthand** (*shorthand properties*). La gran ventaja es que crear reglas con shorthand nos permite escribir un código mucho más corto y más fácil de leer para quienes tienen experiencia en la creación de hojas de estilos.

Para comprender mejor esto, veamos un ejemplo donde se definen los márgenes de manera estándar:

```
margin-top:10px;
margin-bottom:7px;
margin-right:15px;
margin-left:20px;
```

Si buscamos un editor de páginas web WYSIWYG compatible con HTML5 y CSS3, **BlueGriffon** puede ser una buena solución. Esta herramienta es gratis y multiplataforma (Windows, Linux y Mac OS X), y utiliza motor Gecko, al igual que Mozilla Firefox. Encontraremos más información sobre esta herramienta y las opciones de descarga en el sitio web **http://bluegriffon.org**.

Utilizando la propiedad shorthand, que respeta el sentido horario, el código se vería de la siguiente manera:

```
margin:10px 7px 15px 20px;
```

Más adelante veremos otros ejemplos prácticos del uso de esta opción que nos ayuda a ahorrar muchas líneas de código. Podemos conocer más sobre propiedades shorthand en el sitio **www.w3.org/TR/CSS21/about.html#shorthand**.

Herencia, cascada y especif cidad

Ahora que conocemos cómo se construyen las reglas, estamos listos para iniciar la creación de estilos en CSS. Existen varios factores que influyen en el armado de reglas y que debemos tener presentes a la hora de poner manos a la obra con nuestro proyecto.

A continuación, hablaremos sobre tres conceptos fundamentales en el armado de reglas CSS: **herencia**, **cascada** y **especif cidad**.

Herencia

En capítulos anteriores, hemos visto que los elementos HTML cuentan con propiedades por defecto para ser representados en el navegador. Esto quiere decir que, si un elemento no tiene aplicada ninguna regla CSS, por defecto tendrá asignada una forma de mostrarse en el navegador.

EL USO DEL SELECTOR UNIVERSAL

Debemos saber que el símbolo * se encarga de actuar como un selector universal. De todos los selectores posibles que podemos utilizar, este es el menos específico de todos. Se puede emplear en reglas para predefinir valores generales para todos los elementos de la página, aunque luego, al utilizar el selector más específico, los valores aplicados en esta regla serán pisados.

Una buena práctica para lograr compatibilidad entre diferentes navegadores es utilizar **resets** CSS, que permiten establecer valores por defecto. Podemos crear nuestro propio reseteador de estilos o recurrir a los que han armado otros desarrolladores que comparten su código.

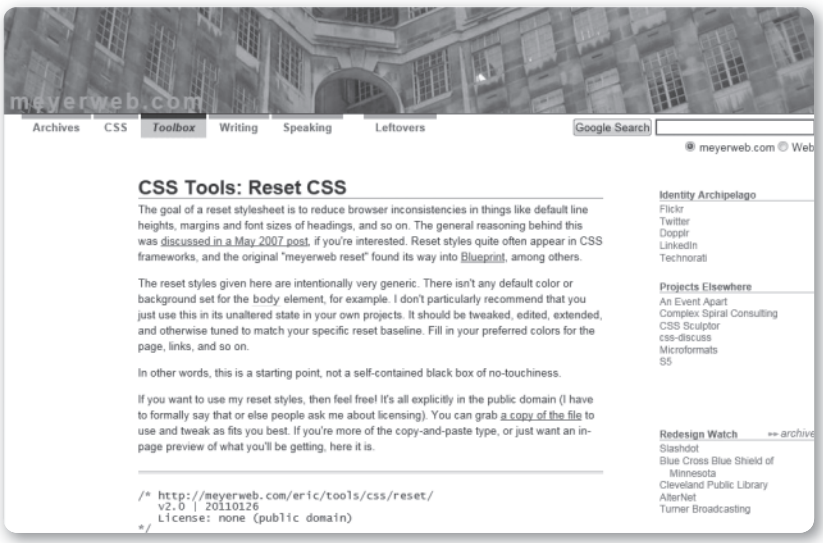


Figura 5. Uno de los resets más famosos es el creado por **`Eric Meyer`**, y podemos observar el código ingresando en **`http://meyerweb.com/eric/tools/css/reset`**.

Si deseamos asegurarnos de que nuestra página se vea de manera correcta en diferentes navegadores y medios, deberemos establecer las reglas que nos garanticen esto. Es necesario tener en cuenta que, en algunos casos, será preciso establecer reglas diferentes para que, en determinados medios (móviles, impresoras, etcétera), el resultado sea adecuado. Para esto, tenemos la regla **@media** que puede recibir los tipos **all**, **braille**, **embossed**, **handheld**, **print**, **projection**, **screen**, **speech**, **tty** y **tv**. Hablaremos más sobre ella en el **Capítulo 7** de este libro, destinado a móviles, en el que también analizaremos las opciones que introduce CSS3 en lo que se refiere a **Media Queries**.

Otro aspecto para considerar es que los elementos pueden **heredar** algunas de las características de su padre. ¿Un ejemplo? Si aplicamos una

regla de estilo de texto, como puede ser el color, a un contenedor ** **, sus párrafos **<p>** hijo que no tengan aplicadas reglas más específicas heredarán esta característica. Observemos la siguiente regla CSS:

```
div {color: red;}
```

Debemos tener en cuenta que si esta regla se aplica al siguiente ejemplo, siempre que no exista otra regla más específica, el párrafo se verá de color rojo y no de color negro, como sería por defecto:

```
<div>
        <p>Texto de mi párrafo.</p>
</div>
```

La herencia puede ser muy útil si se utiliza con criterio. Un aspecto para tener en cuenta es que no toda las propiedades se heredan (podemos analizarlo visitando el sitio que se encuentra en la dirección **www.w3.org/TR/CSS21/propidx.html**).

Cascada y especif cidad

Una importante característica que debemos conocer sobre CSS es cómo actúa la cascada cuando las reglas creadas se superponen.

¿Cómo ocurre esto? Supongamos que, en diferentes partes del código de una página, ya sea en el CSS anexado o en el escrito en el propio documento, escribimos reglas que apuntan al mismo elemento HTML, estableciendo una propiedad con diferentes valores.

¿Qué ocurre en este tipo de choque? ¿Qué regla prima sobre la otra?

En primer lugar, la cascada nos indica que la regla que esté más abajo y que sea igual de específica será la que se aplique. Esto parece muy simple, pero aquí es donde entra a jugar un concepto muy interesante e importante: la **especif cidad**.

En palabras simples, podríamos decir que el motor de render del navegador utiliza este principio para saber qué regla debe utilizar cuando hay propiedades que se superponen.

Para saber el peso que tiene, podemos basarnos en un sistema de puntuación, en el que los elementos valen 1 punto; las clases (**class**), 10 puntos; los identificadores (**id**), 100 puntos, y la declaración de

estilo inline, 1000 puntos. También debemos destacar que hay una declaración que toma mayor peso si se le agrega **!important**.

En síntesis, siempre ganará la regla más específica, y la igualdad de especificidad que triunfará es la que se encuentre más abajo en la cascada. En todos los casos, una declaración inline será la más específica, salvo que otra tenga **!important**.

Estilos con CSS 2.1

Ahora que hemos conocido cómo se utilizan los estilos CSS, la manera en que se construyen reglas y los conceptos de cascada, herencia y especificidad, veremos las principales propiedades que nos ofrece el estándar CSS 2.1 para trabajar.

A continuación, encontraremos un cuadro en el que se muestran las propiedades que se pueden aplicar a las fuentes.

PROPIEDADES PARA LAS FUENTES

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
font	Se utiliza para aplicar diferentes propiedades a la fuente, actuando como shorthand.	Puede recibir los valores de las propiedades **font-style**, **font-variant**, **font-weight**, **font-size**, **line-height**, **font-family** y también **inherit**.
font-style	Estilo de fuente.	**normal**, **italic**, **oblique** e **inherit**.
font-variant	Variante de fuente para indicar cómo se mostrará cuando es mayúscula.	**normal** (valor predeterminado), **small-caps** e **inherit**.

CSS3 Y LA CREATIVIDAD

La llegada de CSS3 abre un camino creativo que puede resultar muy interesante para los diseñadores web. A partir de este nuevo nivel de CSS, es posible imaginar y llevar a la práctica soluciones de diseño que antes requerían el apoyo de otras técnicas que consumían mayor cantidad de recursos.

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
font-weight	Permite definir el peso o grosor de la fuente.	Puede recibir los valores **normal** (predeterminado), **bold**, **bolder** o **lighter**. También se le puede asignar un valor desde 100 a 900 (con pasos de 100 en 100).
font-size	Tamaño de la fuente.	Puede recibir valores en unidades de longitud. También puede recibir valores de tamaños absolutos (**xx-small**, **x-small**, **small**, **medium**, **large**, **x-large** o **xx-large**) o relativos (**larger** o **smaller**), además puede ser **inherit**.
line-height	Permite definir el tamaño del alto de la línea de texto.	Puede recibir valores con unidades de longitud.
font-family	Con esta propiedad, es posible indicar el nombre de una familia de fuente o de la familia genérica.	El valor debe ser el nombre de la familia; se pueden separar por comas los reemplazos y el nombre de la familia genérica. Si el nombre de la familia está formado por más de una palabra, debe ir entre comillas. También puede recibir el valor **inherit**.

Tabla 1. Propiedades CSS 2.1 relacionadas con propiedades de las fuentes.

Como vimos en la tabla anterior, HTML5 nos propone una gran cantidad de opciones para que podamos acceder a trabajar con las propiedades de fuentes en nuestro proyecto web.

POSICIÓN RELATIVA O ABSOLUTA CON CSS

Es interesante tener en cuenta que la propiedad CSS denominada **position** nos entrega la posibilidad de realizar la definición de la posición de un elemento. Para la realización del maquetado, en general será conveniente trabajar con **static** (se trata del valor por defecto que sigue el f ujo normal de los elementos) o también con **relative** (el cual nos permitirá definir una posición relativa al f ujo). Una opción útil para hacer frente a algunas necesidades de diseño es **absolute**, que se encarga de permitir sacar al elemento del f ujo y ubicarlo en el lugar que deseemos.

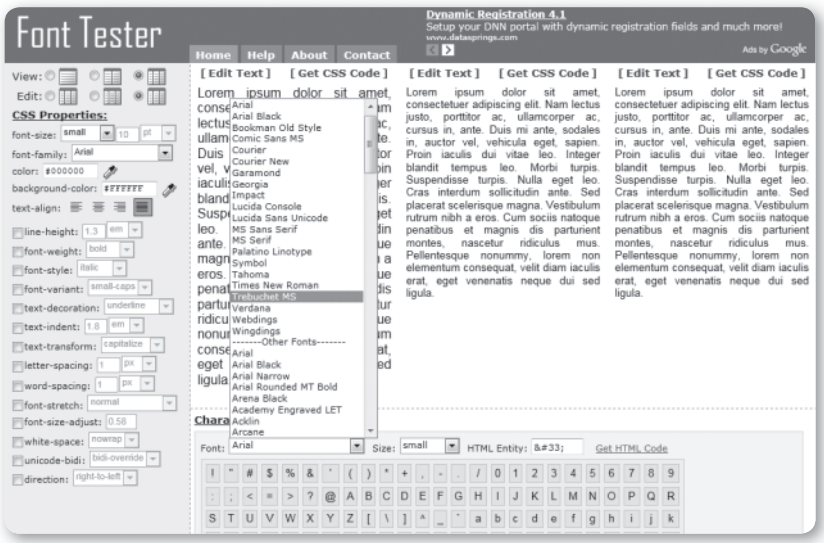


Figura 6. `Font Tester` (`www.fonttester.com`) es una herramienta online muy completa para probar estilos de fuentes y nos ofrece la posibilidad de obtener el código CSS de los estilos que creemos.

En la siguiente tabla, nos encargaremos de analizar aquellas propiedades que se relacionan con las características de texto; por ejemplo definir, la dirección y sangría, entre otras opciones.

PROPIEDADES PARA EL TEXTO

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
word-spacing	Espacio entre palabras.	Puede recibir el valor **normal** (por defecto), una unidad de longitud o **inherit**.
letter-spacing	Espacio entre letras.	Se le puede aplicar el valor **normal** (por defecto), una unidad de longitud o **inherit**.
text-decoration	Decoración del texto.	**none** (por defecto), **underline**, **overline**, **line-through**, **blink** o **inherit**.

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
vertical-align	Permite indicar la alineación vertical del texto.	Puede recibir un valor porcentual, una unidad de medida, o: **baseline**, **sub**, **super**, **top**, **text-top**, **middle**, **bottom**, **text-bottom** o **inherit**.
text-transform	Permite transformar el texto.	**none** (por defecto) **capitalize**, **uppercase**, **lowercase** o **inherit**.
text-align	Indica la alineación del texto.	**left**, **right**, **center**, **justify**, **inherit**.
text-indent	Sangría de texto.	Se le puede aplicar una medida de longitud o un porcentaje, también recibe **inherit**.
direction	Define la dirección de escritura.	Puede recibir los valores **ltr**, **rtl** o **inherit**.
quotes	Permite definir los caracteres que se ubican antes y después de las citas.	Recibe **none**, **inherit** o cadenas de texto con los caracteres que se utilizarán.

Tabla 2. Propiedades CSS 2.1 que se encuentran vinculadas con texto.

Figura 7. `Blueprint` (`www.blueprintcss.org`) es un framework de CSS que promete reducir el tiempo que nos lleva escribir código CSS. Utiliza resets, grids, estilos de formulario y estilos de impresión.

A continuación, veremos una tabla que nos muestra las propiedades relacionadas con colores, fondos e imágenes.

COLORES, FONDOS E IMÁGENES		
PROPIEDAD CSS	**DESCRIPCIÓN**	**VALORES SOPORTADOS**
color	Permite indicar el color de un elemento.	Tengamos en cuenta que los valores que puede recibir están determinados por las unidades de color (por ejemplo, nombre de color, hexadecimal, RGB). También puede recibir el valor denominado **inherit**. A partir de la versión 3 de CSS se suman las posibilidades de RGBA, HSL y HSLA.
background	Puede utilizarse como una propiedad shorthand para aplicar todas las características de fondo.	Debemos saber que esta propiedad puede recibir los valores siguientes: **background-color**, **background-image**, **background-repeat**, **background-attachment**, **background-position** y también es posible utilizar el denominado **inherit**.
background-color	Se utiliza para especificar el color de fondo.	Se encuentra determinado por las unidades de color (al igual que la propiedad llamada **color**). Por defecto es **transparent**. Puede recibir también el valor **inherit**.
background-image	Permite indicar la ruta y el nombre de una imagen de fondo.	Se encarga de recibir la URL de la imagen, como podemos apreciar en el siguiente ejemplo: **url(/img/nombre_de_la_imagen.png)**. También puede recibir el valor **none** (por defecto) e **inherit**.
background-repeat	Se utiliza para especificar cómo se repetirá la imagen de fondo.	Puede recibir los valores **repeat** (valor por defecto de la propiedad), **repeat-x**, **repeat-y**, **no-repeat** o **inherit**.
background-attachment	Se emplea para indicar si la imagen que definimos como fondo queda fija o se mueve cuando nos desplazamos mediante el scroll vertical.	Puede recibir los valores **scroll** (valor predeterminado), **fixed** e **inherit**.

▼ PROPIEDAD CSS	▼ DESCRIPCIÓN	▼ VALORES SOPORTADOS
background-position	Permite que indiquemos la posición inicial que tendrá la imagen de fondo.	Puede recibir unidades de medida o porcentuales (para los ejes X e Y en ambos casos). También puede recibir los valores **left top**, **left center**, **left bottom**, **right top**, **right center**, **right bottom**, **center top**, **center center**, **center bottom** e **inherit**.
clip	Permite indicar la porción de imagen que se mostrará. Es útil para los casos en los que la imagen es más grande que su contenedor.	Puede recibir una forma y su valor es **rect**; entre paréntesis recibe los valores numéricos de los lados (arriba, derecha, abajo e izquierda). También puede recibir los valores **auto** o **inherit**.

Tabla 3. Propiedades CSS 2.1 de colores, fondos e imágenes, además de sus respectivas descripciones y también los correspondientes valores soportados.

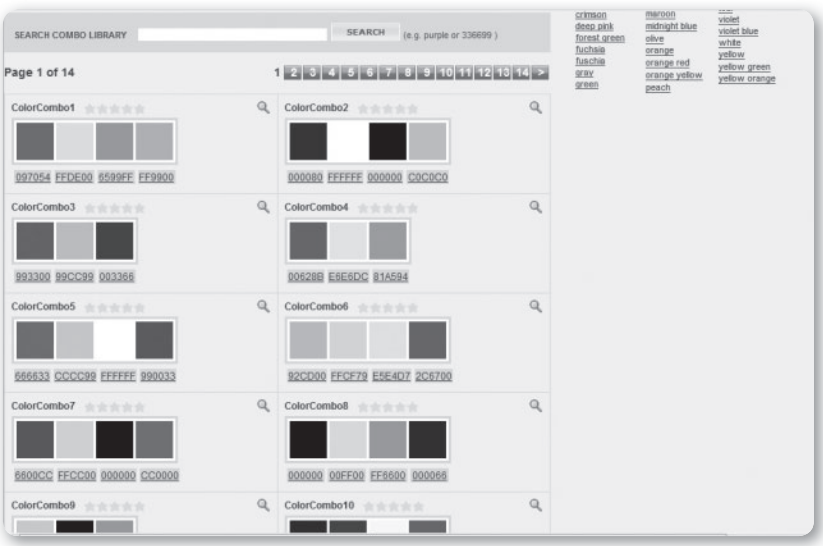


Figura 8. `Color Combos` (`www.colorcombos.com`) es un sitio que nos brinda información sobre paletas de colores. Además, ofrece búsquedas personalizadas y cuenta con herramientas interesantes.

Los colores pueden aplicarse también por su nombre en inglés. Las opciones que comentamos a continuación nos ofrecen una solución sencilla si conocemos el idioma, ya que será fácil identificarlos.

Para esta finalidad tenemos una serie de palabras clave relacionadas que podemos utilizar: **aqua** (celeste), **black** (negro), **blue** (azul), **fuchsia** (fucsia), **gray** (gris), **green** (verde), **lime** (lima o verde claro), **maroon** (marrón), **navy** (azul marino), **olive** (verde oliva), **purple** (púrpura), **red** (rojo), **silver** (plata), **teal** (turquesa), **white** (blanco) y **yellow** (amarillo).

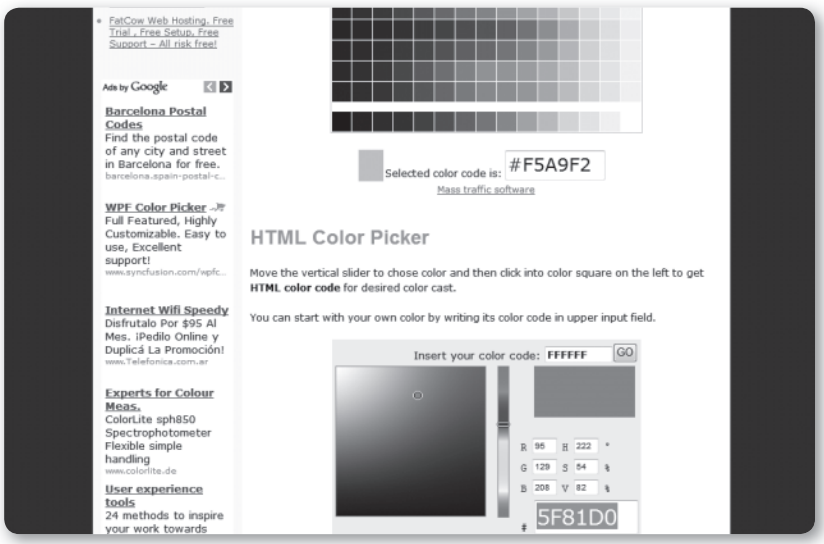


Figura 9. `HTML Color Codes` (`http://html-color-codes.info`) nos brinda una herramienta online que facilita la elección de colores y ofrece su valor resultante en hexadecimal.

LA PALETA DE COLORES

Es interesante tener en cuenta que la paleta de colores es uno de los temas que debemos contemplar al planear el sitio web que deseamos desarrollar. Es importante tenerla definida antes de poner manos a la obra en el diseño. Si no cumplimos esta premisa, es posible que después perdamos mucho tiempo corrigiendo los colores para que respeten la paleta que definamos.

Para acceder a una mayor variedad de colores, podemos utilizar los colores por su valor expresado en hexadecimal. Más adelante en este libro, al hablar de las características de CSS3, veremos también que podremos utilizar colores RGB, RGBA, HSL y HSLA.

En la tabla que veremos a continuación, analizaremos las características de CSS que se encuentran relacionadas con margen (**margin**), relleno (**padding**), borde (**border**), contorno (**outline**), desbordamiento (**overflow**), posición (**position**), índice Z (**z-index**) y espacios en blanco (**white-space**). Veremos la descripción de cada una de estas propiedades así como también los valores soportados.

PROPIEDADES PARA MARGENES Y SIMILARES

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
margin	Se utiliza como propiedad shorthand para establecer los márgenes del elemento.	Puede recibir el valor **auto**, **inherit** o valores relativos o absolutos. Un solo valor aplica igual a todos los lados. Con dos valores, el primero será para superior e inferior, y el segundo, para derecho e izquierdo. Con cuatro valores, aplica el primero al margen superior.
margin-top, **margin-right**, **margin-bottom**, **margin-left**	Márgenes superior, derecho, inferior e izquierdo (se puede utilizar cada uno por separado).	Pueden recibir **auto**, **inherit**, valores de medidas relativas o absolutas.
padding	Esta propiedad se utiliza como shorthand, y permite definir el espacio entre el borde y el contenido del elemento (relleno o padding).	Puede recibir **inherit**, valores de medidas relativas o absolutas. Si recibe un solo valor, aplica igual a todos los lados. Si recibe dos valores, el primero será para superior e inferior, y el segundo, para derecho e izquierdo. Si recibe cuatro valores, aplicará el primero a la parte superior y, luego, seguirá aplicando a cada lado en sentido horario.
padding-top, **padding-right**, **padding-bottom**, **padding-left**	Relleno superior, derecho, inferior e izquierdo (se puede utilizar cada uno por separado).	Pueden recibir **inherit**, valores de longitud o porcentuales.

▼ PROPIEDAD CSS	▼ DESCRIPCIÓN	▼ VALORES SOPORTADOS
Border	Se utiliza para definir las propiedades del borde del elemento.	Recibe los valores de las propiedades **border-width**, **border-style** y **border-color**. También puede recibir el valor **inherit**.
border-top, **border-right**, **border-bottom**, **border-left**	Se utilizan para definir las propiedades de los bordes superior, derecho, inferior e izquierdo del elemento.	Pueden recibir los valores de las propiedades de ancho, estilo y color de cada uno de los bordes de los lados. También pueden recibir el valor **inherit**.
border-width	Ancho del borde del elemento.	Puede recibir los valores **thin**, **medium**, **thick** o una unidad de longitud. Si recibe un solo valor, aplica a todos los lados el mismo. Puede recibir hasta cuatro valores para aplicarle a cada uno de los lados. También puede recibir el valor **inherit**.
border-top-width, **border-right-width**, **border-bottom-width**, **border-left-width**	Ancho del borde superior, derecho, inferior e izquierdo (se puede utilizar cada uno por separado).	Pueden recibir los valores **thin**, **medium**, **thick** o una unidad de medida de longitud. También pueden recibir el valor **inherit**.
border-color	Color del borde del elemento.	Puede recibir como valor entre un color (todos los bordes igual color) y cuatro colores (cada borde un color). También puede recibir los valores **transparent** o **inherit**.
border-style	Estilo del borde del elemento.	Puede recibir los valores **none** (por defecto), **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset**, **outset** o **inherit**. Puede recibir desde un valor (todos los lados el mismo estilo) hasta cuatro (cada lado un estilo distinto).
outline	Permite definir el contorno alrededor del elemento.	Puede recibir los valores de las propiedades **outline-color, outline-style**, **outline-width** o **inherit**.
outline-color	Color del contorno.	Debemos tener en cuenta que puede recibir un color, **invert** o **inherit**.

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
outline-style	Estilo del contorno.	Puede recibir los valores **none** (por defecto), **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset**, **outset** o **inherit**.
outline-width	Ancho del contorno.	Puede recibir los valores **thin**, **medium**, **thick** o una unidad de medida de longitud. Si recibe un solo valor, aplica a todos los lados el mismo. Puede recibir hasta cuatro valores para aplicarle a cada uno de los lados. También puede recibir el valor **inherit**.
overflow	Permite indicar el desbordamiento (overf ow) para el scroll.	Puede recibir los valores **visible** (por defecto), **hidden**, **scroll**, **auto** o **inherit**.
position	Permite definir la posición del elemento.	Puede recibir los valores **static** (por defecto), **relative**, **absolute**, **fixed** o **inherit**.
z-index	Eje Z.	Puede recibir los valores **auto**, **inherit** o bien un número de orden en el apilamiento del eje z (positivo o negativo).
white-space	Espacios en blanco.	Puede recibir los valores **normal** (por defecto), **nowrap**, **pre**, **pre-line**, **pre-wrap** o **inherit**.

Tabla 4. Propiedades de CSS 2.1 vinculadas con margen, relleno, borde, contorno, desbordamiento, índice Z y también espacios en blanco.

Para definir las dimensiones de los elementos, contamos con las propiedades ancho (**width**) y alto (**height**). Para el maquetado de una

THE ART OF WEB

El sitio **The Art of Web** nos ofrece una interesante colección de notas y tutoriales para aprender más sobre el uso de HTML, CSS, JavaScript, SQL y PHP. Esta web se encuentra en idioma inglés, pero ofrece mucha claridad con la ayuda de ejemplos comprensibles e imágenes que ilustran la explicación. El enlace para ingresar a este sitio web es **www.the-art-of-web.com**.

página debemos saber utilizar la flotación (**float**) y el despeje (**clear**). También podemos trabajar con la visibilidad de los elementos con **visibilitiy** y con el modo en que se muestran con **display**.

Cada una de estas propiedades y sus valores correspondientes las analizaremos en la tabla que se muestra a continuación.

DIMENSIONES, FLOTANTES Y VISIBILIDAD

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
width	Ancho del elemento.	Soporta el valor **auto** (por defecto), **inherit** o valores de medida relativos o absolutos.
height	Altura del elemento.	Soporta el valor **auto** (por defecto), **inherit** o valores de medida relativos o absolutos.
float	Permite definir la f otación de un elemento.	Se le pueden asignar los valores **none** (predeterminado), **left**, **right** o **inherit**.
clear	Permite despejar y cortar con la f otación. Puede despejar hacia uno de los lados, hacia ambos o hacia ninguno (por defecto).	Puede recibir los valores **none** (predeterminado), **inherit**, **left**, **right**, o **both**.
visibilitiy	Brinda la posibilidad de especificar la visibilidad del elemento.	Puede recibir los valores **visible** (predeterminado), **inherit**, **hidden** o **collapse**.
display	Permite modificar la manera en que los elementos se mostrarán en pantalla.	**inline**, **none**, **block**, **inline-block**, **inline-table**, **list-item**, **run-in**, **table**, **table-caption**, **table-cell**, **table-column**, **table-column-group**, **table-footer-group**, **table-header-group**, **table-row**, **table-row-group** o **inherit**.

Tabla 5. Propiedades de CSS 2.1 relacionadas con alto, ancho, f otación, despeje de f otantes y también visibilidad.

Las listas son muy utilizadas en diseño web actual para resolver una gran variedad de necesidades. Podemos emplearlas, por ejemplo, dentro de menús de navegación, modificando su aspecto mediante estilos CSS. A continuación, vamos a ver una tabla que nos permitirá analizar las propiedades que se les pueden aplicar a las listas.

PROPIEDADES PARA LISTAS

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
list-style	Funciona como un shorthand de los estilos de lista.	Puede recibir los valores de las propiedades **list-style-type**, **list-style-position** y **list-style-image**. También puede recibir el valor **inherit**.
list-style-type	Permite especificar el tipo de marcador que tendrán los ítems de la lista (solo funciona si el valor de **list-style-image** es **none**, valor por defecto).	Puede recibir los valores **disc** (por defecto), **circle**, **square**, **decimal**, **decimal-leading-zero**, **lower-roman**, **upper-roman**, **lower-greek**, **lower-latin**, **upper-latin**, **armenian**, **georgian**, **lower-alpha**, **upper-alpha**, **none** o **inherit**.
list-style-image	Utiliza una imagen como marcador para los elementos de la lista.	Puede tomar el valor **none**, **inherit** o recibir la URL con la dirección y el nombre de la imagen.
list-style-position	Permite indicar en qué posición se sitúa el marcador respecto del ítem correspondiente de la lista.	Puede recibir los valores **outside** (por defecto), **inside** o **inherit**.

Tabla 6. Propiedades de CSS 2.1 relacionadas con listas.

Para crear una barra de navegación con las listas debemos tener en cuenta que en HTML es necesario definir una **<ul>** y cada ítem del menú lo ubicamos en un **<li>**, En CSS, al elemento li le asignamos la propiedad **list-style-type** con valor none y para que el menú sea horizontal, a los elementos li, en CSS, les asignamos **display:inline**.

PRÁCTICA Y EXPERIMENTACIÓN

Ser un experto en el maquetado de un sitio Web lleva su tiempo. Además de aprender el uso de las etiquetas HTML y los secretos de las hojas de estilo CSS, también es importante dedicarle mucho tiempo a la práctica. Antes de ponernos con un proyecto de manera profesional es recomendable probar y experimentar diferentes opciones para ganara experiencia en el tema.

Figura 10. CSS `Menu Maker` (**`http://cssmenumaker.com`**) es una herramienta online para crear menús para nuestros proyectos web. En pocos pasos, obtenemos un menú personalizado.

PROPIEDAD CSS	DESCRIPCIÓN	VALORES SOPORTADOS
content	Se utiliza junto con los pseudoelementos **:before** y **:after** para insertar contenido.	Puede recibir los valores **normal** (por defecto), **none**, **counter**, **open-quote**, **close-quote**, **no-open-quote**, **no-close-quote** o **inherit**. Con **attr**, recibe entre paréntesis el atributo. Si se utiliza **url**, lo que se define es un contenido multimedia.
caption-side	Se utiliza para especificar dónde se ubicará la leyenda de la tabla.	Puede recibir los valores **top**, **bottom** o **inherit**.

Tabla 7. Otras propiedades útiles que corresponden a la versión 2.1 de CSS.

Si deseamos saber más sobre esta versión, en **www.w3.org/TR/CSS21** encontraremos las últimas novedades de CSS 2.1.

Selectores avanzados

Además de los selectores de elementos que ya hemos conocido, también podemos encontrar otros que nos permitirán realizar operaciones un poco más complejas.

Debemos saber que los **selectores de atributos** nos permiten seleccionar un elemento por su atributo o por el valor del atributo. Para seleccionar un elemento que cuente con un determinado atributo, lo hacemos así: **nombre_elemento [nombre_atributo]**. Si deseamos seleccionar un elemento por el valor de su atributo, lo hacemos de la siguiente forma: **nombre_elemento [nombre_atributo=valor]**.

Para apuntar a un atributo que puede tener múltiples valores, si deseamos especificar que nuestra selección contenga al menos uno de esos valores, podemos utilizar **nombre_elemento [nombre_atributo ~=valor]**.

Para aclarar el tema, veamos un ejemplo en el cual asignamos un **background-color** amarillo a aquellos **<input>** que sean **"type=text"**:

```
input[type="text"] {background-color: yellow;}
```

También existen opciones para seleccionar elementos **adyacentes**, es decir, que tienen el mismo padre. Esto se logra utilizando el signo **+**. La sintaxis para construir la regla es la siguiente: **elemento1 + elemento2**

{propiedad:valor;}. Debemos saber que la regla se aplicará a **elemento2**, siempre que sea adyacente de **elemento1**. Por otro lado, es importante saber que también tenemos la posibilidad de seleccionar elementos que sean hijos directos utilizando el símbolo **>**. La sintaxis que corresponde a la regla es la siguiente: **elemento_padre > elemento_hijo {propiedad:valor;}**. La regla será aplicada a **elemento_hijo** siempre y cuando sea hijo directo de **elemento_padre** (no puede tener otro padre).

Figura 11. Los foros son un buen lugar para compartir experiencia con otros usuarios. **`Foros del Web`** (**`www.forosdelweb.com`**) se especializa en desarrollo web y tiene un subforo especial para CSS.

Pseudoelementos y pseudoclases

Los **pseudoelementos** nos permiten hacer referencia a una parte del elemento y así poder asignarle una característica particular mediante las reglas correspondientes a CSS.

Con **:first-line**, podemos seleccionar la primera línea de un elemento de texto. De esta manera, le podremos aplicar una propiedad diferente de la primera línea, por ejemplo, de un párrafo **<p>**, donde podremos indicar una sangría distinta, un color diferente o alguna transformación de texto.

Con **:first-letter**, tenemos la posibilidad de seleccionar la primera letra de un elemento de texto. En forma adicional, pasarla a mayúsculas o minúsculas, o también cambiar el color correspondiente a este elemento, solo es un ejemplo de algunas de las opciones que podremos aprovechar con esta interesante característica.

Los pseudoelementos **:before** y **:after** nos permiten agregar contenidos y, como veíamos en la **Tabla 7**, trabajan con **content**.

Las **pseudoclases** nos brindan la posibilidad de referirnos a un estado particular del elemento. La forma en que las escribimos es **elemento:pseudoclase**. Su uso más conocido es para los distintos estados de un enlace, donde podemos encontrar las opciones **:link** (estado predeterminado), **:hover** (mouse encima), **:visited** (visitado) y **:active** (activo). También resulta muy útil **:focus** (enfocado), que se utiliza en muchas ocasiones para trabajar con controles de formulario.

Con **:first-child**, podemos seleccionar el primer hijo de un elemento. Otra pseudoclase que puede resultar útil en algunas ocasiones es **:lang**, si lo que necesitamos es seleccionar algún elemento por su idioma.

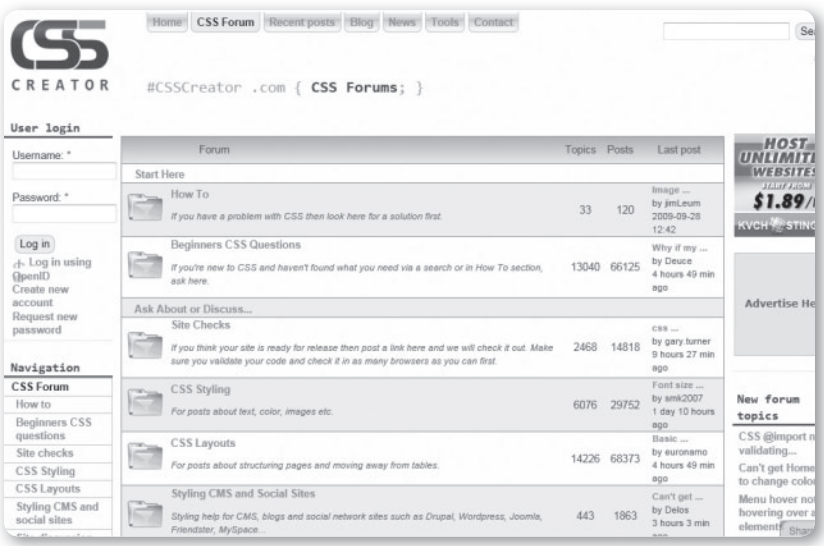


Figura 12. CSS Creator nos ofrece un foro en inglés donde se pueden discutir temas relacionados con CSS (lo encontramos en la dirección **http://csscreator.com/forum**).

Unidades de medida para el desarrollo web

Debemos tener en cuenta que las unidades de medida o de longitud (*length* en inglés) se pueden utilizar para darles valor a algunas de las propiedades de CSS. Por ejemplo, se utilizan para dimensionar un elemento, sus márgenes, relleno, etcétera.

Entre las unidades de medida que podemos utilizar, encontramos las **relativas** y las **absolutas**. Las unidades relativas son **em** (relativa a la altura de la fuente) y **ex** (relativa a la altura de la letra x).

Las unidades absolutas son: **in** (pulgadas), **cm** (centímetros), **mm** (milímetros), **pt** (puntos) y **pc** (picas).

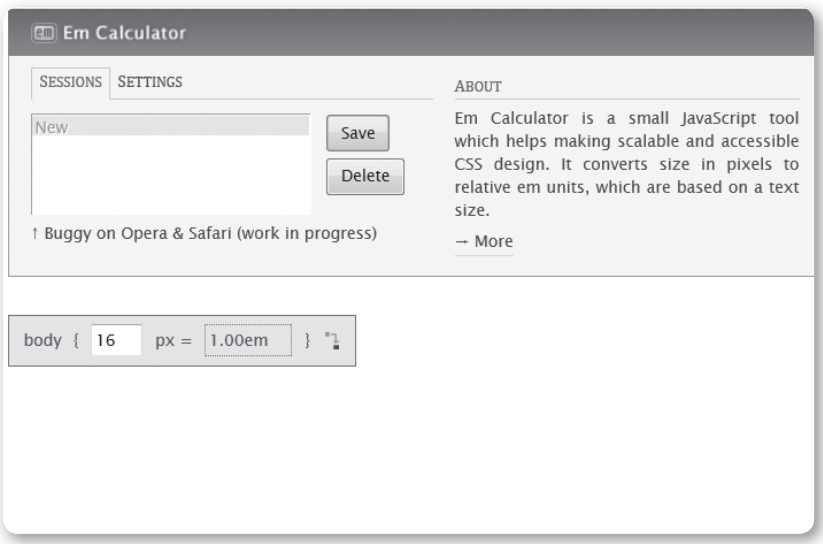


Figura 13. Utilizar medidas **em** es útil si deseamos crear sitios para diferentes resoluciones. **Em Calculator** (**http://riddle.pl/emcalc**) es una herramienta online que nos ayudará con los cálculos.

Un caso para tener en cuenta, ya que resulta ser una unidad muy utilizada, es **px** (píxel). Algunos dan esta unidad como absoluta (1px = 0,75pt), pero hay que tener en cuenta que, en la práctica, resulta relativa a las características de la pantalla o medio del dispositivo.

Es necesario considerar que los valores porcentuales se expresan con el símbolo **%**; son muy útiles, por ejemplo, para definir dimensiones elásticas en el ancho de las columnas y también del contenedor en el maquetado. Algunos diseñadores también han comenzado a utilizar medidas relativas, como el caso de **em**, para lograr "elastizar" y entregar dinamismo a sus diseños.

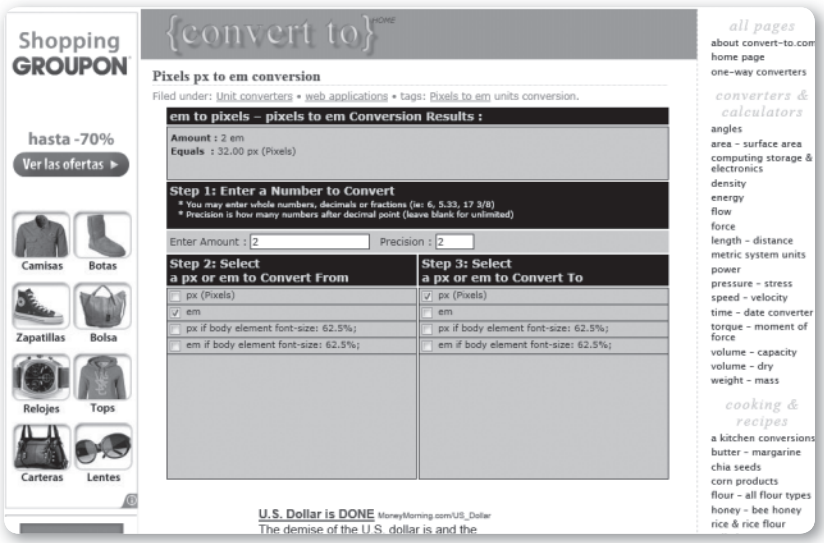


Figura 14. Si ingresamos enen sitio web **`http://convert-to.com/pixels-px-to-em-conversion.html`**, encontraremos una muy buena herramienta para hacer conversiones entre **em** y **px**.

USO DE SPRITES

Es interesante saber que el uso de sprites tiene una larga historia, en especial en el proceso de animación de videojuegos. En la tarea de diseñar sitios web, los sprites nos permiten realizar la inclusión de varios recursos gráficos en una sola imagen; luego, trabajando con la propiedad **background-position** podemos ubicar la porción por mostrar. Esta técnica tiene una amplia aceptación en la actualidad, ya que permite optimizar las transacciones con el servidor; de esta forma, lograremos animaciones sencillas sin necesaidad de utilizar técnicas más complejas.

CSS3: nuevas características

Con la llegada de CSS3, se agregan nuevas características a las que ya conocíamos en CSS 2.1 para lograr efectos y soluciones de diseño sin necesidad de requerir ayuda de otros lenguajes. Este nivel es una evolución muy importante para el estándar; por tal motivo, en las próximas páginas detallaremos las características principales que incorpora. Entre ellas, se destacan las que tienen que ver con sombras (a texto y cajas), bordes (colores, imagen y redondeado), colores y opacidad, trabajo sobre texto, fuentes, pseudoelementos, rotación, propiedades de interfaz de usuario, transformación, transición y animación, entre otras.

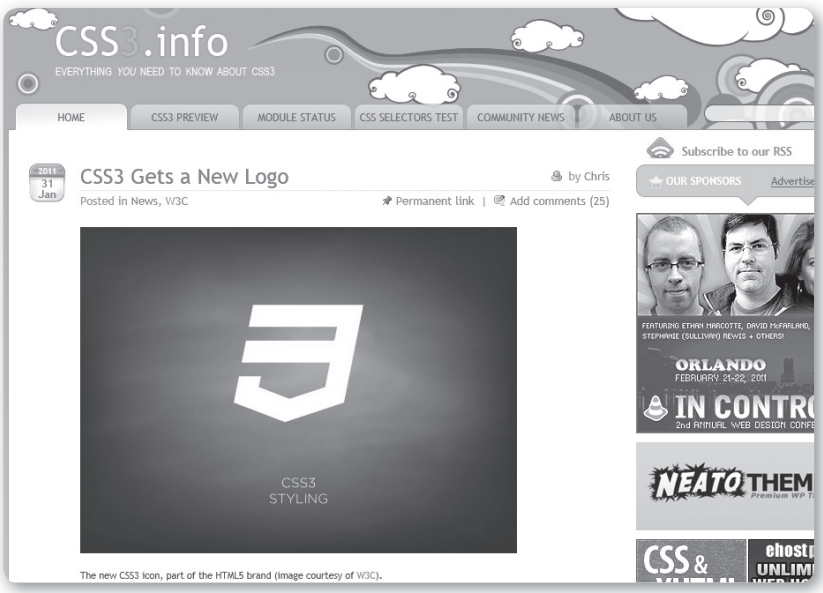


Figura 15. El sitio que encontramos en la dirección **`www.css3.info`** nos ofrece información y noticias sobre CSS3.

Es importante señalar que si bien las aplicaciones que se clasifican como WYSIWYG relacionadas con diseño web han comenzado a incorporar ayuda en el código para el uso de CSS3, podremos encontrar herramientas gratuitas online de gran calidad para ayudarnos a crear los estilos utilizando las nuevas características de este nivel de CSS.

Figura 16. `CSS 3.0 Maker` (`www.css3maker.com`) es una herramienta online que nos permitirá crear estilos y efectos, aprovechando las características de CSS3 de una manera muy intuitiva.

Hasta la estandarización de este nivel, por una cuestión de compatibilidad, algunas propiedades pueden requerir adicionalmente el uso de los prefijos **-moz** (para Mozilla), **-webkit** (para WebKit) y **-o** (para Opera).

Sombra en el texto

CSS3 nos trae dos propiedades especialmente pensadas para aplicar sombras. Una de ellas está relacionada con sombras de texto (**text-shadow**). El modo de uso de **text-shadow** es bastante sencillo. En su declaración, puede recibir cuatro valores: color, coordenada de la sombra respecto al eje X (posición relativa al texto), coordenada de la sombra respecto al eje Y (posición relativa al texto) y radio del blur de la sombra. Vale aclarar que **text-shadow** es una propiedad que se hereda. A continuación veremos un código de ejemplo de un estilo con **text-shadow**:

```
.destacado {text-shadow: 5px 5px 2px #000;}
```

Figura 17. Los efectos de sombra en texto nos entregan la posibilidad de resolver situaciones de diseño que antes requerían del uso de imágenes para su creación.

Si trabajamos con diferentes valores en los ejes y en el blur, podremos personalizar el resultado a nuestro gusto. Incluso, es posible repetir la sombra de un elemento de texto o emplear valores negativos, para lograr darle al texto efectos verdaderamente sorprendentes.

También podemos lograr efectos de luz utilizando colores claros, y no es necesario emplear en todos los casos todas las propiedades, ya que en algunas oportunidades no es preciso aplicar blur; así realizaremos esta tarea de una forma más sencilla. A continuación, veremos el código adecuado para crear un contorno negro en un titular principal cuyo texto es de color blanco.

```
h1 {
color: white;
font-size: 36px;
text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
  }
```

Debemos tener presente que un efecto similar se puede lograr empleando la propiedad denominada **text-outline** (aunque aún se encuentra en discusión si finalmente queda en el estándar). Esta propiedad recibe valores de espesor, blur y color. Veamos un ejemplo:

```
.textoutline {text-outline: 2px 2px #333;}
```

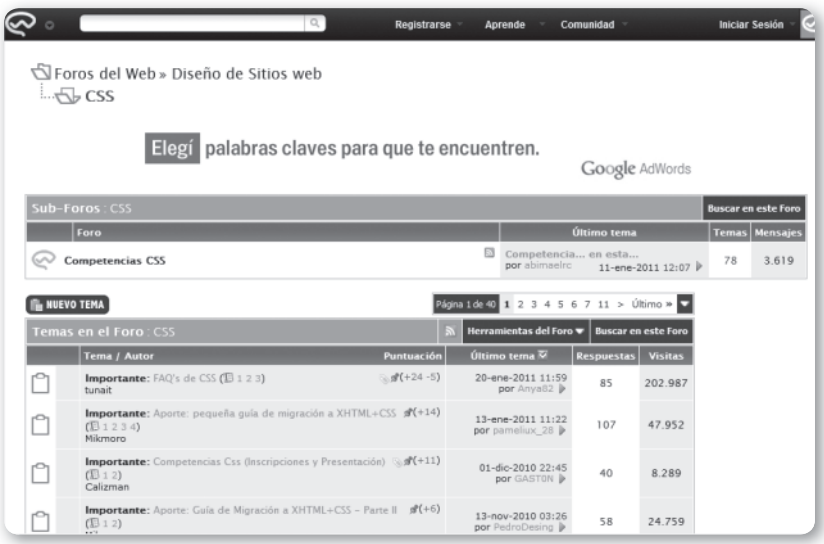


Figura 18. Si ingresamos en el sitio web **`http://westciv.com/tools/shadows`**, encontraremos una herramienta online que nos ayudará a crear efectos con **`text-shadow`**.

CSSBLOG.ES

Una buena alternativa para tener en cuenta, si buscamos información adicional sobre CSS y además nos interesa conocer y acceder a recursos, herramientas y ejemplos de código sobre este tema, es el sitio web que encontramos en la dirección **www.cssblog.es** donde existe una excelente colección de material y también enlaces muy interesantes relacionados con CSS. El sitio se encuentra en idioma español y ofrece contenidos para usuarios de nivel inicial, intermedio y avanzado.

Multicolumna

Con CSS3, llegan novedades para trabajar con múltiples columnas.

La propiedad **column-count** permite indicar en cuántas columnas se dividirá el contenido de un elemento contenedor. El valor que recibe equivale a la cantidad de columnas. Supongamos que contamos con ** ** y deseamos dividir su contenido en cuatro columnas, deberíamos crear una regla como la siguiente:

```
div {column-count:4;}
```

Por compatibilidad, también deberíamos agregar las opciones para Mozilla (**-moz-column-count**) y Webkit (**-webkit-column-count**).

Esta propiedad no se hereda y, además de recibir un número, puede tener el valor **auto**, cuya acción será ajustar el contenido a otra propiedad aplicada (por ejemplo, **column-width**).

Con **column-gap**, especificamos el espacio entre columnas con un valor establecido en una unidad de longitud:

```
column-gap:40px;
```

Esta propiedad no se hereda y, si buscamos compatibilidad con Mozilla y Webkit, deberíamos incluir **-moz-column-gap** y **-webkit-column-gap**.

Para establecer el ancho de las columnas, encontramos la propiedad **column-width**, que puede recibir como valor una medida. Por ejemplo:

```
column-width:100px;
```

El valor por defecto es **auto**, en cuyo caso el navegador determina el ancho. Esta propiedad no se hereda. Si buscamos compatibilidad con Webkit, debemos utilizar también **-webkit-column-width**.

Si deseamos establecer una regla (una línea vertical) entre las columnas, tenemos la propiedad **column-rule**. Podemos utilizar **column-rule-width** (el ancho de la regla), **column-rule-style** (el estilo de la regla, que puede ser **none**, **hidden**, **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset** u **outset**) y **column-rule-color** (el color del separador).

Es interesante saber que estas propiedades no se heredan y pueden ser utilizadas con **-moz** adelante (de esta forma podremos compatibilizar con el navegador Mozilla) y con **-webkit** (para Webkit).

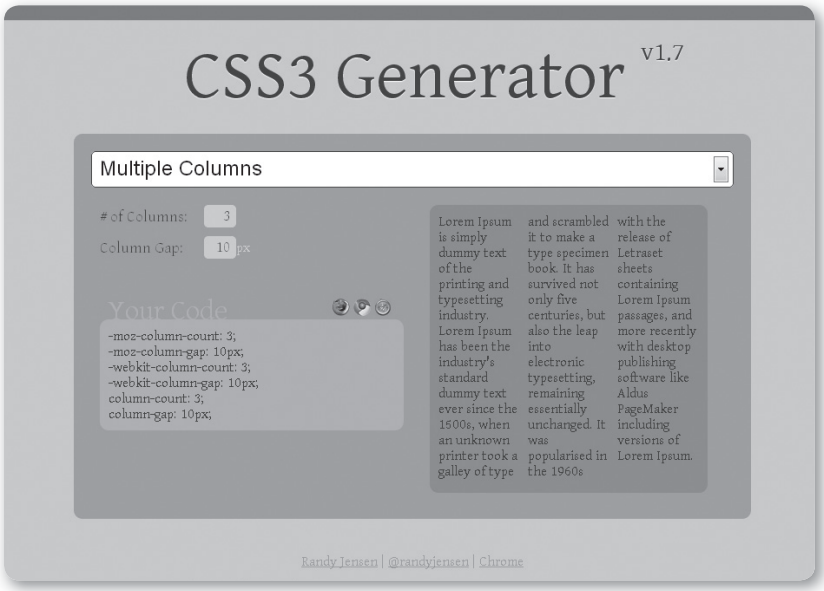


Figura 19. Una forma sencilla de crear reglas multicolumna es hacerlo con **CSS3 Generator** (**http://css3generator.com**).

@fontface

Con la inclusión de esta característica en CSS ahora es posible que accedamos a utilizar fuentes descargables como si se tratara de safe-fonts, así las posibilidades para el diseño aumentan.

Para utilizarla, en primer lugar debemos declarar la regla **@fontface**, especificando nombre de la familia, ubicación y formato de la fuente.

Veamos un ejemplo en el código que se muestra a continuación:

```
@font-face {
    font-family: "Nombre_de_la_familia";
    src: url(./font/fuente.otf) format("opentype");
}
```

Luego, podremos aplicar el nombre de la familia al estilo que deseemos. Veamos un ejemplo aplicado a un párrafo:

```
p {font: 14x "Nombre_de_la_familia", Arial, sans-serif ;}
```

Una cuestión que provoca algunos problemas con el uso de esta característica es que aún no existe un estándar definitivo de fuentes aceptado por todos los navegadores y plataformas. Por tal razón, y para lograr mayor compatibilidad, hay que incluir las fuentes en diferentes formatos, para que los usuarios puedan verlas correctamente.

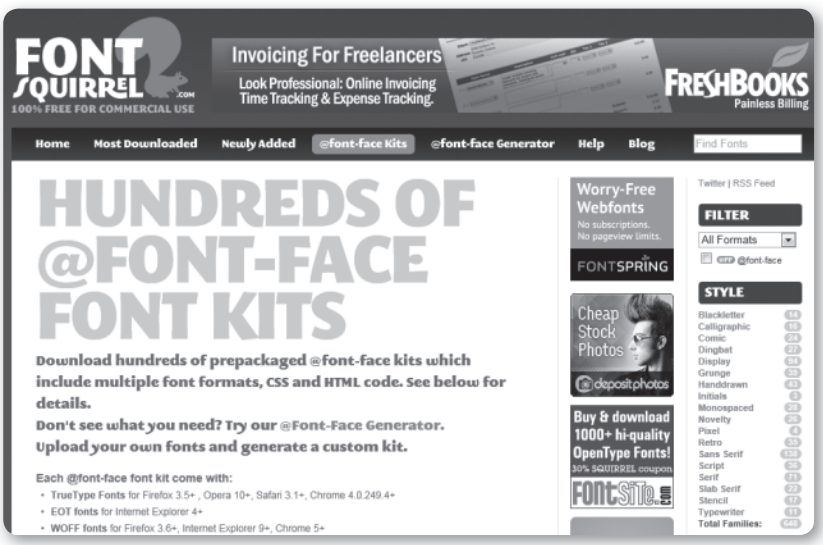


Figura 20. `Font Squirrel` nos ofrece kits de fuentes preparadas para aprovechar las ventajas de **`@fontface`**. La dirección para acceder a ellos es **`www.fontsquirrel.com/fontface`**.

Mientras los navegadores mejoran el soporte para esta característica y se avanza en la definición del estándar definitivo para fuentes en Internet, vamos a repasar los formatos de fuentes que se pueden utilizar con esta característica: TrueType, OpenType, Embedded OpenType y Web Open Font Format (WOFF).

Por sus características de compresión y por estar pensada especialmente para la Web, se espera que WOFF se convierta en el estándar de tipografías para Internet. Este formato se encuentra apoyado por el W3C (**www.w3.org/TR/WOFF**).

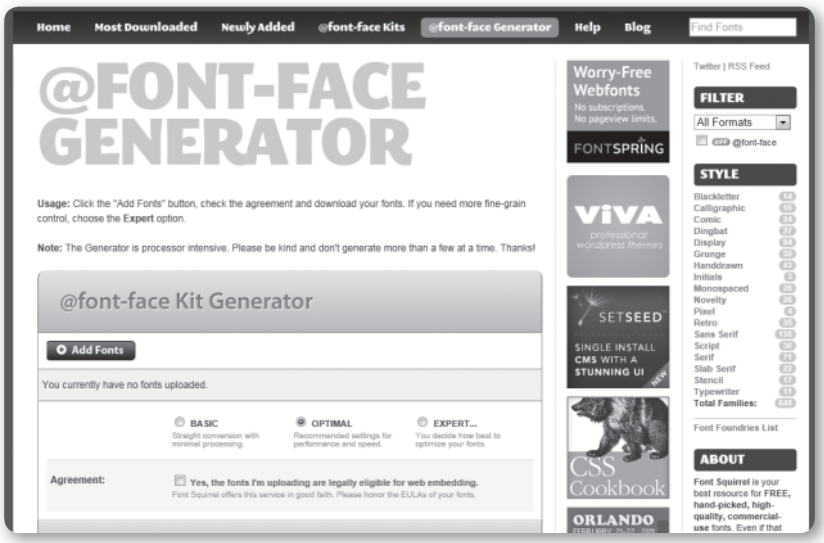


Figura 21. Si necesitamos un conversor para tener fuentes compatibles con distintos navegadores, **`Font Squirrel`** nos lo ofrece en **`www.fontsquirrel.com/fontface/generator`**.

Otras propiedades relacionadas con texto

Además de lo que se refiere a sombreado, CSS3 incorpora varias características relacionadas con texto.

La propiedad **word-wrap** permite definir cómo se actuará ante el corte de palabra. Puede recibir los valores **normal** (la línea se cortará solo en un punto permitido, es decir, no se cortarán las palabras) o **break-word** (las palabras se podrán cortar de manera arbitraria). Esta característica puede ser útil, dependiendo del tamaño de caja que utilicemos para contener el texto. Esta propiedad se hereda.

Por su parte, **text-overflow** es una propiedad que nos permite especificar qué ocurre con el texto cuando llega al límite de su contenedor. Soporta los valores **clip** (valor por defecto, corta el texto), **ellipsis** (muestra **...** para representar que existe más texto disponible) y **string** (muestra una cadena dada para representar el texto cortado). Debemos saber que esta propiedad no se hereda.

Sombreado en la caja

Es necesario tener en cuenta que otra características de sombra que se agrega en CSS3 es la propiedad denominada **box-shadow**. Es interesante saber que se trata de una propiedad que nos permitirá aplicarle sombreado a la caja de un elemento que deseemos.

Figura 22. El efecto **box-shadow** puede crearse con facilidad gracias al uso de **CSS 3.0 Maker**.

TRUCOS PARA LOGRAR UN TEXTO LEGIBLE

Siempre es necesario tener en cuenta algunos tips para lograr que un texto publicado en una página web sea completamente legible para todos los usuarios que la visitan. Entre estas recomendaciones podemos mencionar algunas de las más importantes: en primer lugar, es necesario que la tipografía se muestre en un tamaño no demasiado pequeño; por otro lado, debemos tener mucho cuidado con el contraste que provoca la tipografía con el fondo para no molestar la vista y, por último, no debemos colocar fondos con imágenes que puedan entorpecer la lectura del contenido. Siguiendo estos simples consejos, obtendremos un sitio que atraerá más visitas.

Esta propiedad no se hereda y puede recibir los siguientes valores: **inset** (es opcional y por defecto es **none**, cambia la sombra exterior por una interior), medida de longitud para la distancia horizontal del of set de la sombra, medida de longitud para la distancia vertical del of set de la sombra, medida de distancia del blur, medida de distancia de despliegue y el color para la sombra. Encontraremos más datos y ejemplos de esta propiedad aplicada en el sitio web que está en la dirección **www.w3.org/TR/css3-background/#the-box-shadow**.

Bordes

Dentro de la renovación que trae consigo CSS3, una de las características que mayor utilidad nos ofrece es la de poder trabajar con varias opciones de bordes.

La propiedad **border-radius** permite redondear los lados de la caja del elemento. Puede recibir la medida o un porcentaje. Por ejemplo, para redondear los **<input>**, podemos utilizar el siguiente código:

```
input {border-radius:2em;}
```

Esta propiedad no se hereda y puede utilizarse la forma **-moz-border-radius** para compatibilidad con Mozilla.

Si deseamos asignar diferentes valores a las esquinas, podemos utilizar el shorthand, pasando los cuatro valores separados por espacios a esta propiedad, o bien asignarle valores a cada una con **border-top-left-radius**, **border-top-right-radius**, **border-bottom-right-radius** y **border-bottom-left-radius**.

Aquí vale la pena recordar que, antes del desembarco de CSS3, todo lo relacionado con bordes redondeados debía resolverse con la

OPACIDAD Y TRANSPARENCIA

Los conceptos de **opacidad** y **transparencia** son importantes si deseamos dejar ver total o parcialmente lo que hay detrás de un elemento. Un objeto es completamente opaco cuando no deja pasar la luz y es totalmente transparente cuando la deja pasar por completo. Al trabajar con porcentajes, debemos tener en claro que opacidad y transparencia son conceptos opuestos.

utilización de imágenes que reprodujeran el efecto o con algún script de JavaScript desarrollado para dicho fin.

Otra característica interesante que se incorpora con la versión 3 de CSS es la posibilidad de establecer imágenes para los bordes; para acceder a esta funcionalidad podemos usar **border-image**.

Esta propiedad puede actuar como shorthand para cada uno de los lados. Además, se puede indicar de qué forma se muestra la repetición de la imagen para horizontal y vertical, y soporta: **stretch**, **repeat** o **round**.

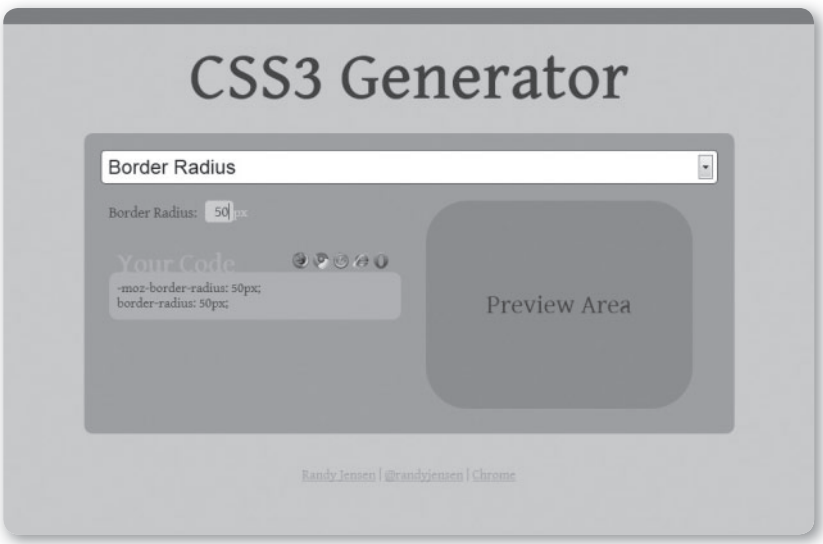


Figura 23. CSS3 Generator es una herramienta online muy completa para generar código CSS3 en pocos pasos y ofrece una opción específica para trabajar con bordes.

CSS EN DIFERENTES MOTORES

El artículo de Wikipedia titulado **Comparison of layout engines (Cascading Style Sheets)**, que podemos consultar en **http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(CSS)**, nos ofrece un muy interesante informe comparativo de las características de CSS en diferentes motores de renderizado, utilizado por los principales navegadores del mercado.

Esta opción puede aplicarse también de manera independiente a los costados (**border-top-image**, **border-right-image**, **border-bottom-image** y **border-left-image**) o a las esquinas (**border-top-left-image**, **border-top-right-image**, **border-bottom-right-image** y **border-bottom-left-image**).

Veamos un ejemplo en el que se aplica a un ** ** una imagen de borde, con el mismo valor a todos los bordes, pero se eligen **repeat** para vertical y **round** para horizontal:

```
div {border-image: url(border.png) 12 round repeat;}
```

Debemos tener en cuenta que esta propiedad puede aplicarse a cualquier elemento y no se hereda. Cabe decir que se trata de una característica que tiene una gran variedad de usos, los cuales van desde crear un marco hasta la posibilidad de diseñar botones, por lo tanto es muy importante tenerla en cuenta.

Para compatibilidad con Mozilla, usamos **-moz-border-image** y para WebKit, **-webkit-border-image**.

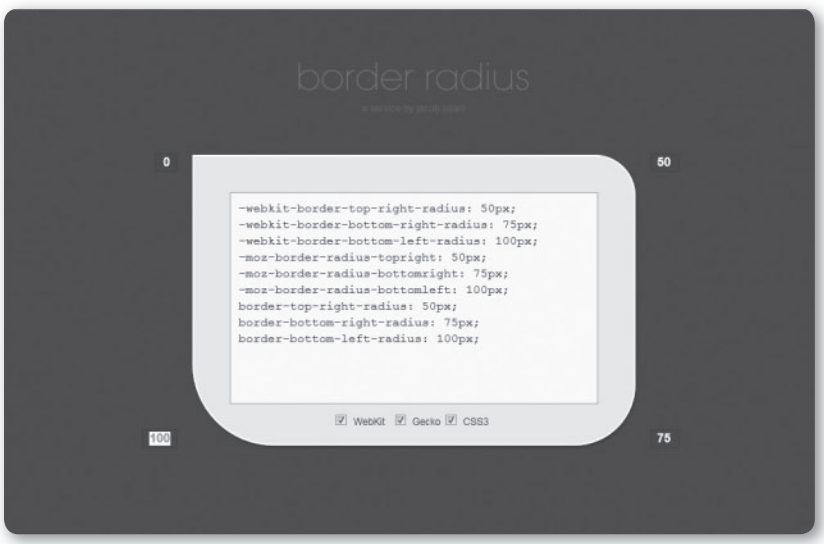


Figura 24. Una solución simple y efectiva para la generación de bordes redondeados está en **`http://border-radius.com`**. Podemos trabajar sobre cada borde por separado y generar código estándar.

Es importante recordar que hay una propiedad que ya conocíamos desde versiones anteriores de CSS y que nos ayudará a la hora de manejar las características del borde. Esta propiedad es la que nos permite definir el grosor del borde (**border-width**). Puede resultar útil incluirla al momento de crear la regla que aplicaremos en el borde.

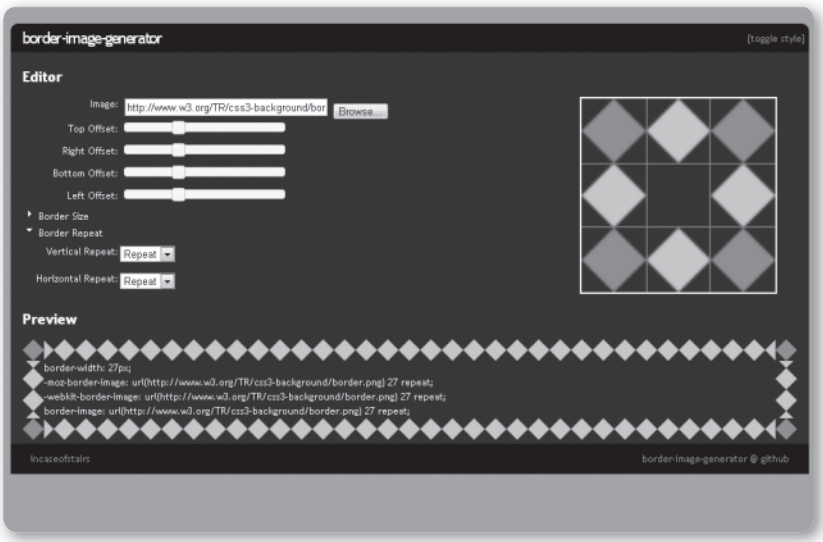


Figura 25. `border-image-generator` (`http://border-image.com`) es una aplicación en línea que simplificará nuestra labor a la hora de trabajar con imágenes para los bordes de los elementos.

Fondos

A partir de CSS3, podemos definir la posición de la imagen de fondo con la propiedad **background-origin**. Esta propiedad puede recibir los valores **padding-box**, **border-box** o **content-box**.

También a partir de este nivel de CSS, se incorpora la posibilidad de especificar el tamaño de la imagen de fondo con **background-size**. Esta propiedad puede recibir una medida fija, porcentual o los valores **cover** (escala la imagen a la menor dimensión para que tanto de ancho como de alto entre en el contenedor) o **contain** (escala la imagen al máximo tamaño para que entre tanto de alto como de ancho en el contenedor).

Para compatibilidad con Mozilla, es necesario agregar **-moz-background**. Una propiedad que ya conocíamos de versiones anteriores es **background-image**, una interesante característica que llega de la mano de CSS3 y da la posibilidad de utilizar múltiples fondos. Su uso es bastante simple; veamos un ejemplo aplicado:

```
div {background-image:url(img1.jpg),url(img2.jpg);}
```

Color

Además de especificar un color por su nombre o su valor hexadecimal, también tenemos la posibilidad de hacerlo mediante su codificación en RGB, RGBA, HSL y también HSLA.

RGB forma los colores con tres valores (rojo, verde y azul). Estos tres valores pueden ir en un rango desde el 0 al 255.

Veamos el ejemplo aplicado a la propiedad color de un párrafo (también se puede aplicar a otros, como borde o fondo en los elementos):

```
p {color: rgb(141,128,85);}
```

Con **RGBA**, tenemos un parámetro más respecto de RGB, y este es el de opacidad, que puede tomar un valor desde 0.0 (totalmente transparente) a 1.0 (completamente opaco). Veamos un ejemplo:

```
div {background-color: rgba(42,152,92,0.4);}
```

OPTIMIZAR EL RENDIMIENTO DEL SITIO

Tengamos en cuenta que aunque hoy en día las conexiones de banda ancha son cada vez más usuales, nunca debemos olvidarnos de la importancia de optimizar la performance de nuestro sitio, pensando en ofrecer velocidades de carga aceptables para todos los usuarios, independientemente del tipo de conexión que tenga cada uno. Las nuevas capacidades que encontramos en CSS3 nos ofrecen diversas soluciones que se encargan de consumir una menor cantidad de recursos y que ya podemos comenzar a aprovechar en nuestros diseños web, para obtener mayores velocidades de carga.

Otra opción para definir el color es mediante el modelo **HSL** (tonalidad, saturación y luminancia). De esta manera, se puede crear un color mediante estos tres valores. La tonalidad (hue) puede recibir un valor de 0 a 360; la saturación (saturation) y la luminosidad (lightness), un porcentaje del 0% al 100%, cada una. A continuación veamos un ejemplo de color HSL aplicado a una clase:

```
.textocolor {color:hsl(320,50%,75%);}
```

Si ingresamos en el sitio web que se encuentra en la dirección **www.w3.org/TR/2003/CR-css3-color-20030514/#hsl-color**, encontraremos una muy buena guía sobre cómo se crean los colores con HSL.

Con **HSLA**, se agrega un parámetro más respecto de HSL, relacionado con la opacidad. Se le puede asignar un valor desde 0.0 (totalmente transparente) hasta 1.0 (completamente opaco). En el código que se muestra a continuación podemos observar un ejemplo:

```
div {background-color: hsla(305,25%,100%,0.3);}
```

Debemos saber que también es posible que accedamos a trabajar de manera directa sobre la opacidad de los elementos con **opacity**. Esta propiedad puede tomar valores desde 0.0 (totalmente transparente) hasta 1.0 (que corresponde a completamente opaco).

Para compatibilidad con la familia de Internet Explorer, podemos agregar el filtro Alpha. Este filtro toma el valor del 0 (transparente) al 100 (opaco). Un ejemplo: **filter:Alpha(opacity=25);**.

Gradient

Así como creamos colores con la versión 3 de CSS, ahora podemos evitarnos el uso de imágenes y crear degradados (gradient) de color, directamente con propiedades de la hoja de estilo.

Debemos tener en cuenta que es posible que realicemos la creación de un degradado lineal con **linear-gradient**, la cual puede recibir los valores de punto de inicio, ángulo y los distintos pasos o paradas de color. Para el navegador Mozilla, usamos **-moz-linear-gradient** y para WebKit, **-webkit-gradient** (debemos especificar tipo **lineal**).

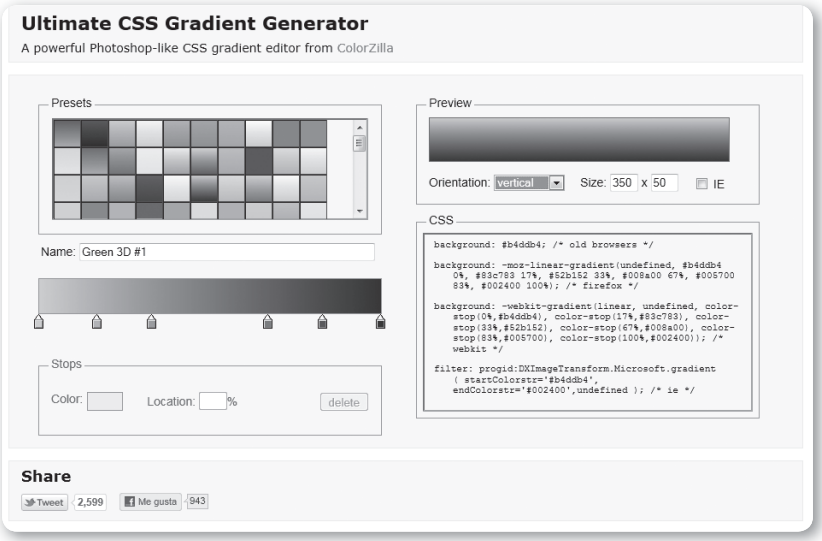


Figura 26. `Ultimate CSS Gradient Generator` es una herramienta que nos facilitará el trabajo con gradientes en CSS. Su dirección es `www.colorzilla.com/gradient-editor`.

Si deseamos crear un degradado radial, usamos **radial-gradient**. Esta recibe la posición, el ángulo, la forma, el tamaño y los pasos o paradas de color. Para Mozilla, usamos **-moz-radial-gradient** y para WebKit, **-webkit-gradient** (debemos especificar tipo **radial**).

Esta característica podemos aplicarla como si fuera un color a la propiedad de CSS3 que deseemos.

Diseños flexibles con el módulo Flexible Box Layout

Lograr un diseño flexible, para que se adapte a las características de la pantalla del dispositivo con el que accede el usuario, es una necesidad frecuente en la creación de sitios web. De la mano de CSS3, se plantean nuevas soluciones para este problema.

Con el módulo **Flexible Box Layout**, se introduce la posibilidad de distribuir de una manera más eficiente y controlada el lugar que pueden

tener asignado los elementos hijo (flexbox items) de una caja que los contiene (flexbox). De esta manera, pueden distribuirse y flexibilizarse los elementos contenidos de acuerdo con las necesidades del diseño.

La propiedad **flex-direction**, aplicable al flexbox, permite indicar cómo se colocarán los elementos hijo dentro de su padre. Los valores que puede recibir esta propiedad son: **inline** (predeterminado), **lr**, **rl**, **tb**, **bt**, **inline-reverse**, **block** o también **block-reverse**.

La propiedad **flex-order** se puede emplear sobre los flexbox ítems para indicar el orden del elemento dentro del grupo. Puede recibir un valor numérico que inicia con el **1**.

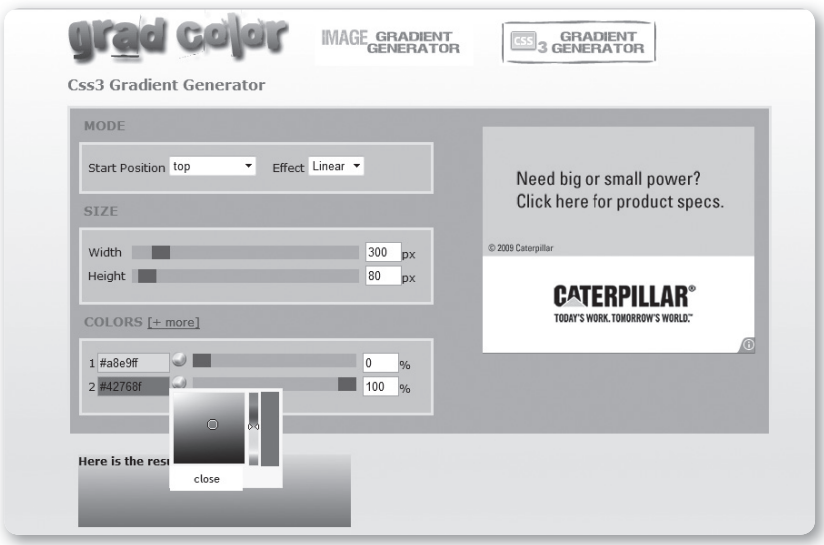


Figura 27. `Grad Color` es un generador de gradientes muy completo. Además de generar la regla CSS3, ofrece el código con el filtro para IE. Su sitio es: **`http://gradcolor.com/css3-gradient.php`**.

Es necesario tener en cuenta que la propiedad denominada **flex-pack** se encarga de permitirnos realizar la definición de cómo se mostrará el espacio flexible alrededor de los flexbox ítems. Esta trabaja con **start**, **end**, **center** y **justify**. Por su parte, **flex-align** permite definir la alineación respecto del espacio libre entre los elementos y puede recibir los valores **auto** (predeterminado) o **baseline**.

Este modelo ya ha comenzado a ser soportado por navegadores con motor Gecko (utilizando **-moz-** para Firefox) y WebKit (con **-webkit-** para Safari y Chrome). En el caso de Microsoft Internet Explorer, lo incluye a partir de la versión 10.

Podremos aprender más sobre Flexible Box Layout Module ingresando en **www.w3.org/TR/css3-flexbox**.

Si deseamos conocer las actualizaciones que está recibiendo este módulo en el Draft podemos visitar el sitio web que encontramos en la dirección: **http://dev.w3.org/csswg/css3-flexbox**.

Propiedades de interfaz de usuario

CSS3 incorpora propiedades relacionados con la interfaz de usuario. En el caso de **appearance**, se puede utilizar para hacer que un elemento luzca diferente de la representación estándar de la interfaz del usuario. Sus valores son: **normal**, **icon**, **window**, **button**, **menú** o **field**. Para Mozilla, usams **-moz-appearance** y para WebKit, **-webkit-appearance**.

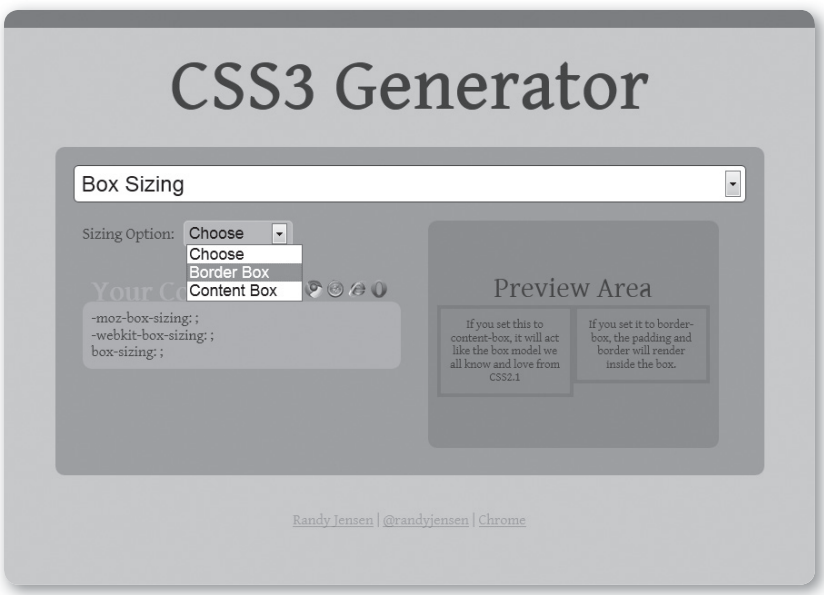


Figura 28. La propiedad **`box-sizing`** puede manejarse con gran facilidad mediante la aplicación online CSS3 Generator.

Con **box-sizing**, podemos indicar de manera precisa cómo una caja llenará un área donde es contenida. Los valores que puede recibir son **content-box** (predeterminado), **border-box** o **inherit**. Para Mozilla, podemos usar **-moz-box-sizing** y para WebKit, **-webkit-box-sizing**.

La propiedad **resize** permite especificar si el elemento puede ser redimensionado por el usuario. Puede recibir los valores **none** (por defecto), **both**, **horizontal** o también **vertical**.

Estas son recomendaciones candidatas para CSS3, y encontraremos más información en **www.w3.org/TR/css3-ui**.

Selectores, pseudoelementos y pseudoclases en CS3

Como veíamos anteriormente, CSS nos permite trabajar con selectores, pseudoelementos y pseudoclases. Con el nivel 3 de este estándar, se agregan nuevas opciones en este campo, y se nos abre un amplio abanico de posibilidades, que nos ofrecen variadas soluciones de diseño sin necesidad de recurrir a JavaScript.

Con este nivel, se agregan nuevos selectores de atributos: **^** (el valor del atributo comienza con), **$** (el valor del atributo termina con) y ***** (el valor del atributo contiene). Para tener un ejemplo, podríamos aplicar una regla de estilo aplicando el color rojo a los enlaces que apunten a una imagen de extensión .JPG. Veamos el ejemplo:

```
a[href$=".jpg"] {color:red;}
```

Además del selector adyacente que conocíamos desde CSS 2.1, ahora encontramos un selector para todos los elementos hermano: **~**. El modo

En el blog **Vagabundia**, que se encuentra en la dirección web **http://vagabundia.blogspot.com**, es posible encontrar una gran cantidad de recursos y también ejemplos aplicados de uso de código HTML, CSS y JavaScript. Encontraremos explicaciones muy didácticas junto a cada recurso disponible; este blog brinda soluciones creativas para diseñadores y desarrolladores web.

de construcción es simple: **elemento1 ~ elemento2 {propiedad:valor;}**. El valor de la propiedad se aplicará a **elemento2**, siempre que sea hermano y aparezca después de **elemento1**, aunque en este caso no necesariamente a continuación.

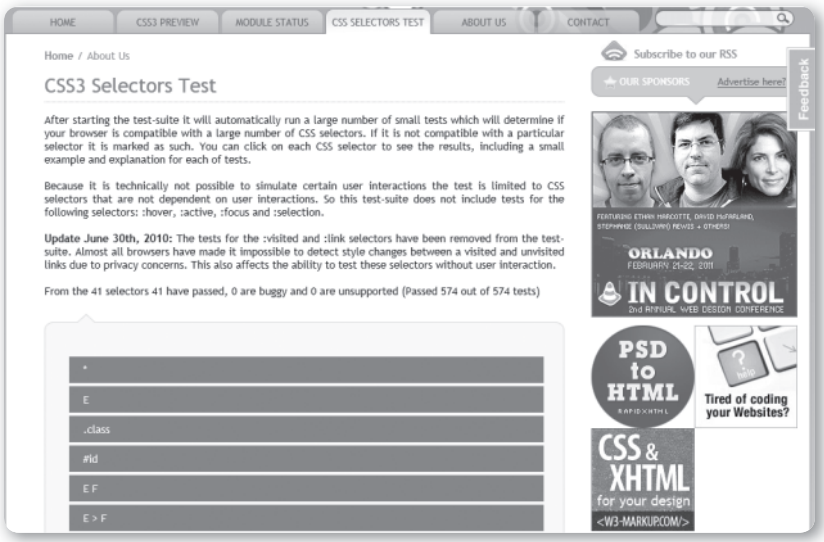


Figura 29. CSS3 `Selectors Test` (`http://tools.css3.info/selectors-test/test.html`) es una herramienta que nos permitirá verificar la compatibilidad con los selectores de CSS3.

Ahora en CSS3, los pseudoelementos deben llevar **::**. También encontramos los conocidos con anterioridad: **::first-line** (primera línea), **::first-letter** (primera letra), **::before** (antes), **::after** (después) y también el que se agrega en este nuevo nivel, **::selection** (para aplicar reglas a la selección que haya realizado el usuario).

Las características más jugosas nos llegan por el lado de las pseudoclases que se incorporan con CSS3, entre las que encontramos:

- **:nth-child()**: permite seleccionar al hijo "n", donde "n" es un número que pasamos entre paréntesis a esta pseudoclase.
- **:nth-last-child()**: similar a la anterior; permite seleccionar al hijo "n", donde "n" es un número que recibe entre paréntesis. La diferencia es que se comienza a contar desde abajo.

- **:empty**: esta característica nos permite seleccionar un elemento que no tenga ningún hijo ni ningún contenido de texto.
- **:first-child**: adecuada para elegir el primer hijo.
- **:last-child**: permite acceder a seleccionar el último hijo.
- **:nth-of-type()**: recibe entre paréntesis un valor numérico "n", que indica qué número de elemento hermano es el del elemento indicado.
- **:nth-last-of-type()**: es una característica igual a la anterior, pero comienza a contar desde abajo hacia arriba.
- **:not()**: nos permite realizar la selección de los elementos que no cumplen con una condición adecuada.

Figura 30. El resultado del código CSS, aplicado a una estructura de tabla de HTML, es el que vemos en esta imagen.

Para visualizar un uso práctico de estas opciones, podemos imaginar la necesidad de pintar las filas de una tabla como "cebra", a dos colores alternados como muestra la imagen anterior.

Ahora bien, ¿qué código será necesario para lograr el efecto de la figura anterior? En primer lugar deberemos crear en HTML una tabla con sus respectivas celdas y contenido. Luego estaremos listos para

crear los estilos. Tengamos en cuenta que el código CSS para tener una tabla con filas verdes con texto plata, y alternarlas con filas marrones y texto blanco es el que vemos a continuación:

```
table {
background: green;
color: silver;
}
tr:nth-child(2n+1) {
background-color: brown;
color:white
}
```

Encontraremos más información sobre estos temas en la siguiente dirección web: **www.w3.org/TR/css3-selectors**.

Transformación

La propiedad denominada **transformation** es una de las más sorprendentes entre las que se incluyen en CSS3, ya que nos permite transformar elementos en 2D y también en 3D.

Por defecto, el valor de esta propiedad corresponde a **none**, pero puede trabajar con una gran variedad de funciones de transformación, entre las que encontramos las siguientes:

- **matrix**: se trata de una función que permite definir una transformación 2D recibiendo una matriz de 6 valores.
- **matrix3d**: permite definir una transformación 3D recibiendo una matriz de 4x4 (16 valores en total).
- **translate**: esta función nos permite definir una traslación 2D; recibe las coordenadas de los ejes X e Y.
- **translate3d**: se puede utilizar para definir una traslación 3D; recibe las coordenadas de los ejes X, Y, Z.
- **translateX**: es una función que podemos emplear para definir una traslación en el eje X; recibe solo ese valor.
- **translateY**: se emplea para acceder a una traslación en el eje Y; debemos saber que recibe solo ese valor.
- **translateZ**: se trata de una función que emplea para definir una traslación en el eje Z; recibe solo ese valor.

- **scale**: se usa para escalar un elemento en 2D; recibe valores de X e Y.
- **scale3d**: escala un elemento en 3D; recibe valores de X, Y, Z.
- **scaleX**: permite definir una transformación en escala dada por un valor del eje X que le pasamos.
- **scaleY**: permite definir una transformación en escala dada por un valor del eje Y que le pasamos.
- **scaleZ**: permite definir una transformación en escala dada por un valor del eje X que le pasamos.
- **rotate**: se usa para rotar el elemento en 2D; recibe un valor angular.

Figura 31. Combinando efectos de transformación de CSS3 con frameworks de AJAX, por ejemplo jQuery, podemos lograr resultados muy interesantes: **`http://css-tricks.com/examples/CSS3Clock`**.

- **rotate3d**: se utiliza para realizar la rotación del elemento en 3D; recibe los valores de X, Y, Z y el ángulo.
- **rotateX**: permite definir una rotación en 3D a lo largo del eje X.
- **rotateY**: permite definir una rotación en 3D a lo largo del eje Y.
- **rotateZ**: permite definir una rotación en 3D a lo largo del eje Z.
- **skew**: recibe los valores del ángulo X y el ángulo Y. Permite "torcer" el elemento en 2D en esos dos ejes.

- **skewX**: se encarga de recibir el valor del ángulo X. Nos permite "torcer" el elemento en 2D en ese eje.
- **skewY**: esta función recibe el valor del ángulo Y. También permite "torcer" el elemento en 2D en ese eje.
- **perspective**: recibe un solo valor y permite definir una perspectiva para un elemento transformado en 3D.

En todos los casos, estas funciones reciben los valores entre paréntesis. Cuando la función recibe más de un valor, estos se separan por comas, siempre dentro del paréntesis.

Figura 32. Las técnicas de animación de CSS3 permiten realizar trabajos de muy buena calidad. Un ejemplo se encuentra en **`http://neography.com/experiment/circles/solarsystem`**.

Podremos encontrar más información sobre las opciones de transformación 2D que ofrece CSS3 en el documento **www.w3.org/TR/css3-2d-transforms**. Para saber más sobre las transformaciones 3D, podemos leer **www.w3.org/TR/css3-3d-transforms**.

Para transformaciones 2D, Mozilla soporta **-moz-transform** y Opera, **-o-transform**. Para transformaciones 2D y 3D, WebKit soporta la opción

-webkit-transform. A continuación, veremos un ejemplo sencillo de una rotación aplicada a una clase **box**, que incluye la compatibilidad para los diferentes navegadores:

```
.box {
        transform: rotate(10deg);
        -moz-transform: rotate(10deg);
        -webkit-transform: rotate(10deg);
        -o-transform: rotate(10deg);
}
```

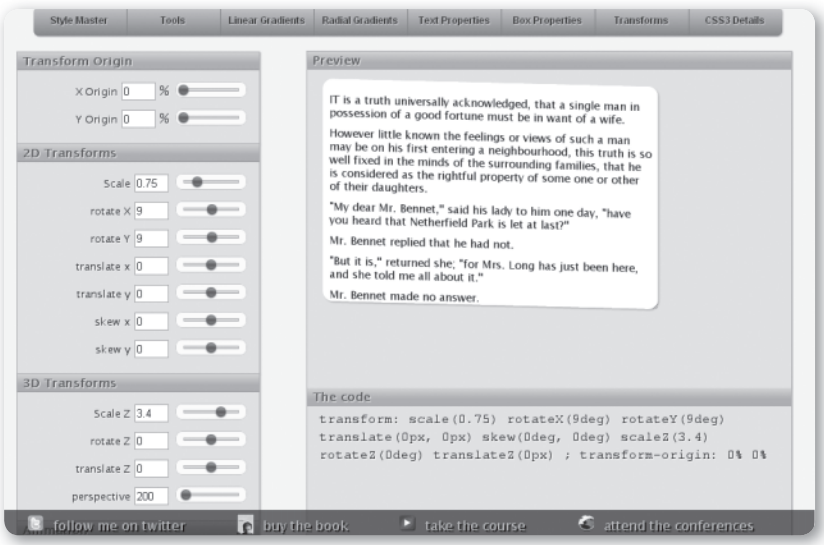


Figura 33. Ingresando en el sitio web **`http://westciv.com/tools/3Dtransforms/index.html`**, encontraremos una excelente herramienta online para crear transformaciones 2D y 3D.

Para la familia de navegadores de IE que no son compatibles con las características de CSS3, podemos utilizar un filtro para realizar operaciones de rotación. Encontraremos más información en **http://msdn.microsoft.com/en-us/library/bb554293(v=vs.85).aspx**.

También hallaremos otros efectos de transformación para la familia Internet Explorer con el filtro de matrix (Matrix Filter), sobre el cual podremos leer más en la dirección web **http://msdn.microsoft.com/en-us/library/ms533014(v=vs.85).aspx**.

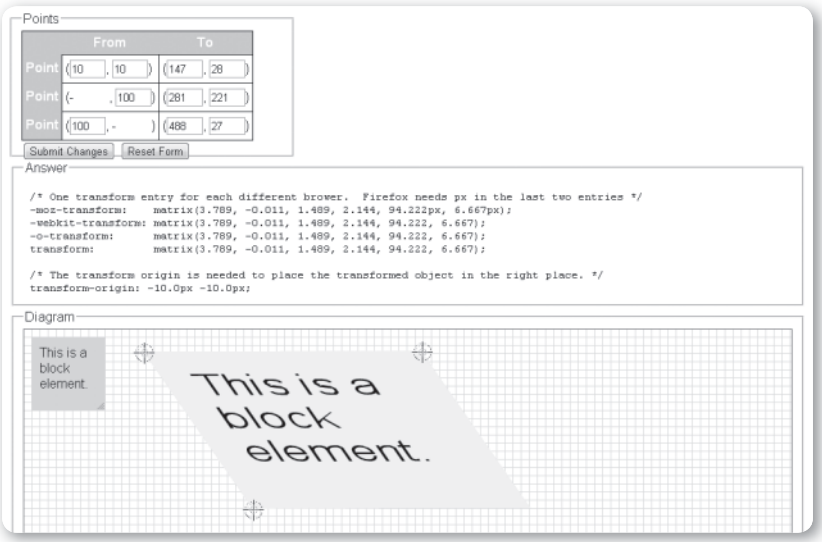


Figura 34. The `Matrix Construction` (`www.useragentman.com/matrix`) es una herramienta online que nos permitirá comprender cómo trabajar con matrices y transformaciones CSS3.

Transición

La propiedad **transition** le otorga a CSS3 una característica visual que permite lograr mejores animaciones empleando estilos. Básicamente, lo que nos ofrece esta característica es la posibilidad de establecer una propiedad cuando un elemento pasa de un estilo a otro. Un ejemplo simple de esto lo encontramos cuando pasamos el mouse sobre un enlace; si tenemos definido que cuando ocurra esa acción el enlace cambie de color, podemos establecer un efecto para esa transición.

Debemos saber que se puede utilizar **-webkit-transition** para navegadores basados en WebKit y **-o-transition** para Opera. Es importante destacar que **transition** funciona como una propiedad

shorthand que puede recibir los valores de las propiedades **transition-property** (nombre de la propiedad a la que se aplicará la transición), **transition-duration** (duración de la transición), **transition-timing-function** (curva de transición del efecto) y **transition-delay** (permite indicar la demora o el momento en que la transición comienza).

Ahora, comenzaremos a detallar cada una de las propiedades de transición. En el caso de **transition-property**, la transición puede recibir los valores **none** o **all** (predeterminado). También es posible indicar, separando por comas, cada una de las propiedades a las que se le desea aplicar la transición correspondiente.

La propiedad denominada **transition-duration** se emplea para indicar el tiempo que durará la transición, y puede recibir el valor expresado en segundos (**s**) o también milisegundos (**ms**).

Figura 35. Para crear transiciones de manera sencilla, es recomendable utilizar la opción **`CSS Transition`** de **`CSS 3.0 Maker`**.

Las transiciones no siempre tienen el mismo tiempo de comienzo y final; para establecer esto, podemos utilizar la propiedad **transition-timing-function**. Esta puede recibir los valores **linear**, **ease**, **ease-in**,

ease-out y **ease-in-out**. También se puede utilizar **cubic-bezier** y pasarle, entre paréntesis, cuatro valores que pueden ser decimales de 0 a 1. Por ejemplo, **ease** equivale a **cubic-bezier(0.25,0.1,0.25,1)**. La ventaja de trabajar con los valores predefinidos es que estos nos permiten generar efectos ya conocidos y estándares; por su parte, hacerlo con **cubic-bezier** nos habilita más posibilidades y la capacidad de tener más control sobre esta interesante característica.

La propiedad **transition-delay** por defecto tiene un valor **0** y puede recibir valores expresados en segundos (**s**) o milisegundos (**ms**).

Los efectos de transición tienen muchas aplicaciones, pero, además de emplearse para animaciones complejas, en el diseño web general pueden ser muy útiles para aplicarlas a enlaces, menús o incluso los famosos acordeones.

Opciones de animación

Las opciones específicas de animación que incorpora CSS3 le permiten, al diseñador, modificar las propiedades de estilos en el tiempo.

Con la regla **@keyframes**, podremos especificar cómo la animación irá cambiando. Al definir la regla, debemos darle un nombre y, luego, determinaremos cómo se modifican las propiedades y los valores en el tiempo, indicando en porcentaje los avances.

Veamos una regla para mover un elemento:

```
@keyframes mover {
  0%  {top: 10px;}
  40% {top: 15px;}
  60% {top: 45px;}
  100% {top: 95px;}
}
```

Es importante mencionar que para usar con WebKit, se puede utilizar la regla conocida como **@-webkit-keyframes**.

Para animación, también se agrega la propiedad shorthand **animation**, la cual se encarga de recibir los valores de las propiedades **animation-name**, **animation-duration**, **animation-timing-function**, **animation-delay**, **animation-iteration-count** y **animation-direction**.

La propiedad **animation-name** permite indicar el nombre de la animación para la regla **@keyframes**. Con **animation-duration**, podremos indicar el tiempo de la animación en segundos. Si utilizamos **animation-timing-function**, podremos indicar la curva de la animación. El valor por defecto es **ease**, pero también podremos aplicar **ease-in**, **ease-out**, **ease-in-out**, **linear** y **cubic-bezier**. Con la propiedad **animation-delay**, podremos especificar el tiempo de demora en segundos. Si empleamos **animation-iteration-count**, podremos especificar la cantidad de iteraciones con un número o con el valor **infinite** (infinito). La propiedad **animation-direction** nos ofrece la posibilidad de indicar si la animación se ejecutará también en reversa. Por defecto, su valor es **normal** (sin reversa). Si aplicamos el valor **alternate**, la animación se correrá normalmente, luego en reversa y, después, comenzará a alternar.

Tengamos en cuenta que en todos los casos, para compatibilidad con WebKit, podremos agregar el prefijo **-webkit-**.

Encontraremos más información y algunos ejemplos sobre el uso de estas propiedades en: **www.w3.org/TR/css3-animations**.

RESUMEN

En el cuarto capítulo del presente libro, aprendimos a trabajar con hojas de estilo CSS 2.1 y CSS3. Analizamos cómo se emplean selectores, clases, pseudoclases, pseudolementos y propiedades. Profundizamos sobre los aspectos principales que hay que saber para construir reglas en CSS. Nos detuvimos en el significado de la cascada en CSS y conocimos el rol fundamental que tiene la especificidad cuando las reglas establecidas en los estilos colisionan. Al detallar las propiedades de CSS3, también vimos en qué casos se requiere agregar prefijos para lograr compatibilidad con los diversos navegadores.

Actividades

TEST DE AUTOEVALUACIÓN

1	¿Qué son los selectores?
2	¿Qué diferencias hay entre usar `id` o `class`?
3	Explique cómo inf uye la especificidad cuando colisionan reglas en CSS.
4	Indique qué propiedades se le pueden aplicar a un elemento de texto en CSS 2.1
5	¿Qué propiedades incluye CSS3 para trabajar con multicolumna?
6	¿Qué opciones encontramos en CSS3 para personalizar los bordes?
7	¿De qué manera se pueden aplicar múltiples fondos con CSS3?
8	¿Cómo se puede trabajar con gradiente en CSS3?
9	¿Qué pseudoelementos y pseudoclases se incorporan a partir de CSS3?
10	¿Cuáles son las ventajas que aportan rotación, transformación y transición al incorporarse al estándar en CSS3?

ACTIVIDADES PRÁCTICAS

1	Cree una regla para definir el color y el tamaño de todos los párrafos de una página.
2	Aplique color de fondo a una celda con CSS 2.1.
3	Aplique una sombra a un texto con CSS3.
4	Cree una regla con @fontface utilizando una tipografía Open Type.
5	Aplique el efecto de transición al texto de un enlace que cambia de color cuando el mouse pasa por encima.

*05 –

Multimedia

En este capítulo, nos introduciremos en las principales características multimedia que ofrece HTML5 y las analizaremos a fondo. Conoceremos las posibilidades que nos brindan y cómo podemos utilizar las etiquetas de audio y video, que se incorporan a partir de HTML5. Además, veremos cómo aplicarlas en nuestros proyectos web.

- **Evolución del concepto multimedia en la Web 186**
- **Incorporar audio con HTML5 ..... 191**
  - La etiqueta <audio> 191
  - Códecs de audio 192
- **Incorporar video con HTML5 ..... 193**
  - La etiqueta <video> 194
  - Códecs de video 195
  - El elemento track 197
- **La etiqueta <source> 198**
- **Aspectos de compatibilidad de audio y video 199**
- **Incorporar los elementos multimedia utilizando HTML5 en nuestro desarrollo 202**
  - Potenciar y personalizar nuestro desarrollo con frameworks de JavaScript 205
- **Acceso a dispositivos multimedia 208**
- **Resumen 209**
- **Actividades 210**

Evolución del concepto multimedia en la Web

En sus primeros años de vida, allá por la década del noventa, la Web no tenía la belleza que le conocemos hoy. No contaba con todo el factor social, ni mucho menos con el look **multimedia** de hoy.

Para comprender mejor esta evolución, es importante recordar que el concepto de multimedia se refiere a una manera de manifestar o transmitir información y contenidos, aprovechando los múltiples medios que tenemos disponibles, entre los que podemos citar audio, video, texto e imágenes. Al hablar de medios electrónicos, muchas veces el término multimedia también llega asociado a otros, como por ejemplo, **interactividad**, que es la posibilidad de que el usuario actúe con el medio y no sea solo un mero espectador que recibe contenidos.

Figura 1. El concepto multimedia excede el ámbito de la computadora y llega a diversos dispositivos, incluyendo el gran boom de los portátiles.

Con la evolución en las técnicas de desarrollo, el aumento en la velocidad de conexión y otros avances relacionados, se ha logrado que las posibilidades multimedia de la Web se extiendan y lleguen a un público cada vez más masivo y demandante.

En lo que se refiere a audio y video, si bien a través de las etiquetas **<object>** y **<embed>** de HTML es posible embeber archivos de audio y video, esta opción puede encontrar algunos problemas de compatibilidad en las diferentes plataformas y navegadores.

Una manera simple de incluir un archivo de audio MP3 utilizando el estándar HTML4 sería la que vemos a continuación:

```
<object>
        <param name="src" value="./audio/pista01.mp3">
        <param name="controller" value="true">
        <embed src="./audio/pista01.mp3"controller="true" ></embed>
</object>
```

Ahora analizaremos un ejemplo de cómo se inserta un video con la extensión **.MOV**. Para hacer más completo el caso, indicaremos que se utilice el reproductor oficial de Apple:

```
<object classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
   id="pelicula"
   width="640" height="480"
codebase="http://www.apple.com/qtactivex/qtplugin.cab">
      <param name="src" value="./vid/mi_video.mov"/>
      <param name="autoplay" value="false"/>
      <param name="controller" value="true"/>

    <embed src="./vid/mi_video.mov" width="640" height="480"
      name="pelicula" autoplay="false" controller="true"
      enablejavascript="true"
       pluginspage="http://www.apple.com/quicktime/download/"/>
    </object>
```

Como podemos observar en el código, resulta necesario utilizar tanto las etiquetas **<object>** como **<embed>** para lograr mayor compatibilidad. Estas etiquetas pueden recibir varios parámetros, como el **classid** (identificador del reproductor), **src** (dirección del video) y otros relacionados con la reproducción y los controles. En el caso de **codebase** para **<object>** y de **pluginpage** para **<enbed>**, se ofrecen enlaces hacia la página que contiene el plugin o complemento para los navegadores que no lo tienen incorporado. En este ejemplo, la referencia es al reproductor de Apple, pero podría tratarse de otro reproductor. El problema que se puede plantear es el relacionado con el soporte para diferentes plataformas que pueda tener dicho reproductor o plugin.

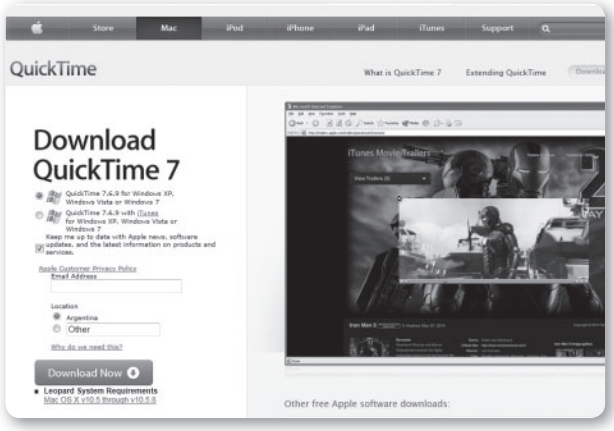


Figura 2. **QuickTime** es una aplicación gratuita que podemos descargar de **www.apple.com/quicktime/download**.

En los últimos años, fue creciendo la aceptación de utilizar reproductores Flash para soluciones web. ¿Por qué? Principalmente porque el agregado de Flash Player está instalado en la mayoría de los equipos, y esto ofrece una buena solución de compatibilidad.

Debemos saber que los principales sistemas de video online se preocuparon de adherir a esta posibilidad.

Figura 3. En el sitio **www.adobe.com/es/products/flashplayer**, podremos ver las ventajas de **Flash Player** y descargarlo.

Cabe destacar que algunos de los navegadores pueden contener este agregado incluido de manera predeterminada, como en el caso de Google Chrome, pero, en la mayoría, será necesario descargarlo.

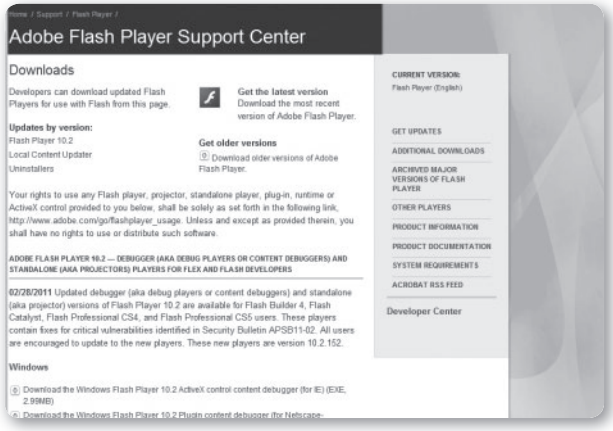


Figura 4. En **`www.adobe.com/support/flashplayer/downloads.html`**, hay datos sobre soporte de Flash Player para desarrolladores.

Es necesario tener en cuenta que uno de los sitios que utiliza reproductores de video basados en Flash es **YouTube** (el cual podemos encontrar en la dirección **www.youtube.com**), el cual ha logrado posicionarse como líder en la tarea de compartir videos online. La evolución de este servicio y la que también ha experimentado Internet junto al video digital permite que hoy sea posible compartir contenidos de video tanto en calidad estándar como en HD.

Una de las grandes ventajas que nos brinda YouTube, entre las que le han permitido extenderse hasta límites inimaginables, es la posibilidad de compartir los videos incrustándolos en otros sitios.

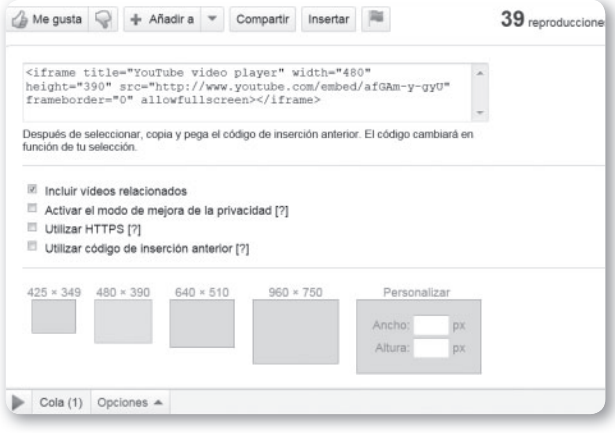


Figura 5. YouTube permite personalizar el reproductor de video al obtener el código para incorporar el video en otros sitios web.

Para citar un ejemplo, si alguien desea subir un *demo reel* de sus trabajos a YouTube, no solo podrá gozar del tráfico que reciba directamente del sitio, sino que, a su vez, podrá tomar el código e incorporarlo en su sitio. Esto le brindará una solución cómoda, ya que no deberá implementar ningún reproductor especial para esta necesidad y, además, no comsunirá recursos del límite de transferencia de su plan, cosa que si ocurriría si tuviera el video alojado en su hosting.

HTML5 PLANTEA UN CAMBIO MUY INTERESANTE PARA LOS REPRODUCTORES MULTIMEDIA DE LA WEB ACTUAL.

Más allá de esta evolución, desde hace tiempo se esperaba un recurso para dar soporte a contenidos multimedia por medio de HTML y que no dependiera de plugins. Como hemos mencionado en capítulos anteriores, HTML5 ofrece una alternativa para estas necesidades, de la mano de etiquetas específicas para audio, video y también para el acceso a dispositivos, otra alternativa que se encontraba monopolizada, principalmente, por Flash.

Muchos sistemas, como Vimeo (**http://vimeo.com**), se dedican a realizar pruebas y migraciones con sus reproductores en búsqueda de la forma de dar soporte a video aprovechando las características de HTML5.

Figura 6. Si deseamos probar el reproductor de HTML5 de YouTube, debemos ingresar en **`www.youtube.com/html5`** y activarlo.

Esto es parte de las tendencias que se viven en Internet con la llegada de HTML5. A continuación, veremos las características más importantes a nivel de multimedia que nos ofrece esta nueva versión del lenguaje.

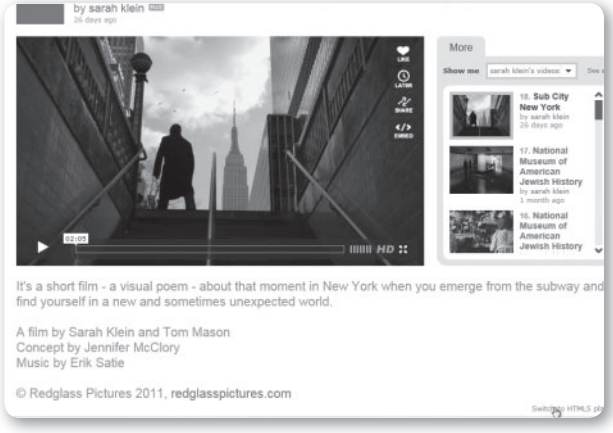


Figura 7. Vimeo cuenta con una comunidad de usuarios registrados y ofrece un servicio de gran calidad; además ofrece la opción de HTML5.

Incorporar audio con HTML5

Los pedidos de los usuarios y desarrolladores han sido oídos y el soporte nativo de audio llega con HTML5. Hace algunos años, la incorporación de audio debíamos realizarla mediante algún tipo de agregado o plugin, para que el usuario pudiera disfrutar esta característica. Esto cambia en HTML5 con la etiqueta **<audio>**.

La etiqueta <audio>

A través de **<audio>**, podremos comenzar a utilizar una de las posibilidades multimedia más importantes de HTML5. Con ella, es posible incluir archivos o streaming de audio con soporte nativo.

NACIMIENTO DE YOUTUBE

Sabemos que YouTube (**www.youtube.com**) es uno de los sitios más populares para compartir videos online y cuenta con millones de visitas diarias de usuarios de todo el mundo. El sitio fue fundado por Chad Hurley, Steve Chen y Jawed Karim en el año 2005. Al año siguiente, y luego de un gran crecimiento, Google se interesó y adquirió el sitio en una cifra millonaria.

Esta etiqueta soporta los siguientes atributos:

- **autoplay**: este atributo se utiliza para indicar si el audio comienza luego de cargar y estar listo.
- **controls**: permite indicar si los controles del reproductor se mostrarán.
- **loop**: se emplea para indicar si el audio comenzará de nuevo una vez que haya terminado de sonar.
- **preload**: permite indicar si el audio se cargará con la página. Puede tomar los valores **auto**, **metadata** y **none**.
- **src**: se trata de un atributo que nos brinda la posibilidad de indicar la dirección URL del recurso de audio.

Esta etiqueta también soporta los atributos globales que mencionamos ahora: **accesskey**, **class**, **contenteditable**, **contextmenu**, **dir**, **draggable**, **dropzone**, **hidden**, **id**, **lang**, **spellcheck**, **style**, **tabindex** y **title**.

Además, vale la pena recordar que tanto para la etiqueta de audio como para la de video nos serán muy útiles los eventos multimedia que se incorporan con HTML5 y que hemos visto en el **Capítulo 2** de este libro. Ellos son los que mencionamos ahora: **oncanplay**, **oncanplaythrough**, **ondurationchange**, **onemptied**, **onended**, **onerror**, **onloadeddata**, **onloadedmetadata**, **onloadstart**, **onpause**, **onplay**, **onplaying** y **onprogress**.

Códecs de audio

Los **códecs** son los que permiten codificar y decodificar información de audio y video que se encuentra en un archivo contenedor.

Esto posibilita, principalmente, tener archivos más livianos con pérdidas (lossless) o sin ellas, hecho que quedó demostrado con el éxito (y la polémica) del MP3 para los archivos de audio.

HISTORIA DEL MP3

Debemos tener en cuenta que la popularidad del formato de audio MP3 es muy conocida, pero por otro lado, la mayoría de los usuarios que lo utilizan a diario no saben mucho de su historia. Tengamos en cuenta que varias patentes relacionadas con esta tecnología se registraron entre mediados de la década del ochenta y principios de los noventa. Pero es interesante saber que recién a mediados de la década del noventa se aplicó realmente en un archivo de computadora, gracias a la labor de Karlheinz Brandenburg; de esta forma nació lo que hoy usamos y conocemos como MP3.

Vale decir que el contenedor de audio puede incluir tanto la pista de audio codificada como información adicional de metadata (dependiendo de cada formato). Los formatos de audio que pueden soportar los navegadores son Ogg Vorbis, MP3 y WAV. Más adelante, en este capítulo, hablaremos sobre el tema compatibilidad.

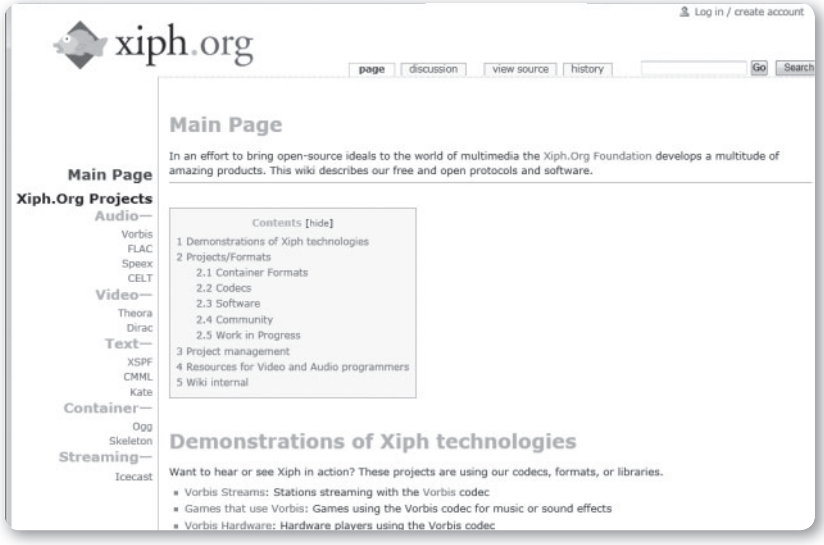


Figura 8. En la dirección **`http://wiki.xiph.org/Main_Page`**, encontraremos una completa wiki donde se describen software, formatos y protocolos libres para recursos multimedia.

Incorporar video con HTML5

De igual manera que la incorporación de la etiqueta de audio, el soporte nativo para el video es otra de las ventajas de HTML5. La necesidad de incluir esta característica ya llevaba largo tiempo de espera y, con su llegada, se abren posibilidades para la implementación de soluciones que requieren mostrar video dentro del navegador.

La etiqueta que posibilita la incorporación de esta característica es **<video>**, y hablaremos sobre ella a continuación.

Figura 9. Podemos probar un video en HTML5 ingresando en **`www.dailymotion.com/html5`**. También tenemos la posibilidad de cambiar el video y ver los cuadros por segundo, entre otras opciones.

La etiqueta <video>

La etiqueta **<video>** se incorpora en HTML5 para poder definir un área donde se mostrará un video. En dicho espacio, se puede incluir tanto un clip como un streaming.

Los atributos que podremos utilizar con la etiqueta **<video>** son los que conoceremos a continuación:

- **audio**: se trata de un atributo que nos permite indicar el estado del audio del video correspondiente. Y, si está disponible la opción, se puede asignar el valor denominado **muted**.
- **autoplay**: se utiliza para indicar que el video comience la reproducción una vez que esté cargado y listo.
- **controls**: podemos emplear este atributo para indicar si deseamos mostrar los controles del reproductor.
- **height**: permite indicar el alto del reproductor.
- **width**: nos da la posibilidad de indicar el ancho del reproductor.
- **loop**: permite indicar que cuando el video termine, haga un loop, es decir, que vuelva a comenzar.

- **poster**: nos brinda la posibilidad de indicar la URL de una imagen que representa al video. En general, puede utilizarse un fotograma o un póster que represente al clip de video.
- **preload**: se utiliza para indicar que el video cargue junto con la página y que esté listo para iniciarse.
- **src**: permite especificar la URL del video.

La etiqueta de video también soporta los atributos globales, al igual que **<audio>** y, además, se puede utilizar con los eventos multimedia mencionados anteriormente.

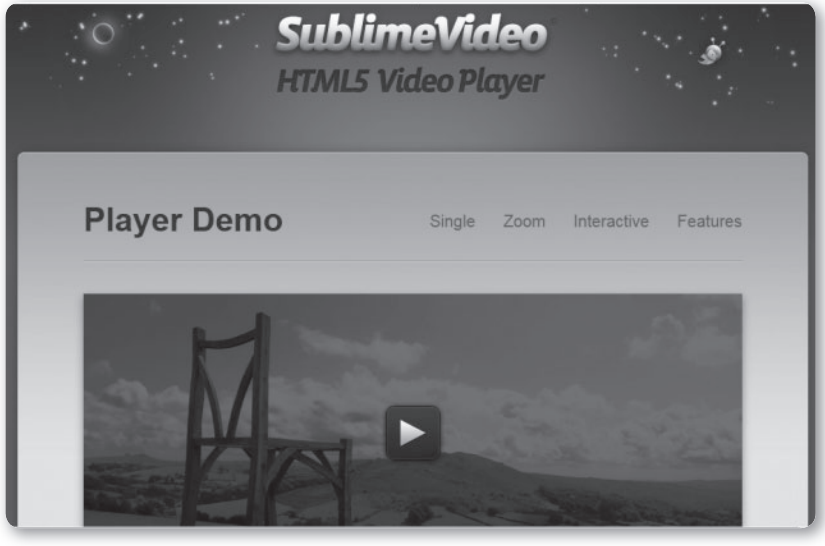


Figura 10. Más adelante veremos cómo incorporar video en nuestro proyecto y qué beneficios nos ofrecen algunos reproductores. Un ejemplo es **`SublimeVideo`** (**`http://sublimevideo.net/demo`**).

Códecs de video

Al hablar de audio, explicamos qué son los códecs. Para el caso del video, debemos saber que el archivo contenedor lleva dentro las pistas de audio y video (codificadas con sus respectivos códecs) y, además, puede llevar consigo información de metadatos.

Para trabajar con las opciones de video de HTML5 en la Web, podemos utilizar los siguientes formatos y códecs:

- Ogg con códec de video Theora y códec de audio Vorbis, aunque esta opción parece haber perdido algo de terreno frente a sus competidores WebM y MPEG4 con H.264.

Figura 11. En el sitio **`www.theora.org`** podremos leer más información sobre **Theora** y acceder a las opciones de descarga.

- WebM con códec de video VP8 y códec de audio Vorbis. Alternativa libre a partir de la iniciativa de Google.

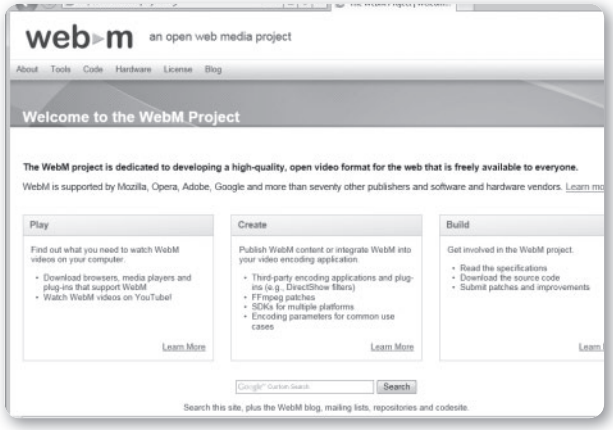


Figura 12. Encontraremos más información sobre el proyecto **`WebM`** en **`www.webmproject.org`**.

- MPEG4 con códec de video H.264 y códec de audio AAC. Debemos tener en cuenta que se trata de una opción que ofrece alta calidad, pero con el uso de un códec propietario.

Figura 13. El códec **H.264** no solo es una solución para la Web, también ofrece una buena calidad de video para otras necesidades.

El elemento track

Aún en desarrollo y evolución, **<track>** está pensado para permitir la inclusión de pistas de texto en los elementos multimedia de HTML5.

Tengamos en cuenta que además de soportar los atributos globales de HTML5, también acepta los atributos siguientes:

- **kind**: es el tipo de atributo, que puede recibir los valores **subtitles** (subtítulos), **caption** (transcripción de diálogos, sonidos, etcétera), **descriptions** (descripción del componente multimedia), **chapters** (títulos del capítulo) y **metadata** (metadatos).
- **src**: permite indicar la URL del track de texto o metadatos.
- **srclang**: permite indicar el lenguaje del track.
- **label**: brinda la posibilidad de incluir una etiqueta para el track.
- **default**: se utiliza para indicar el modo por defecto.

También nos será de utilidad poder acceder directamente a la API de **Text track** (**www.w3.org/TR/html5/video.html#text-track-api**).

FLASH VIDEO (FLV)

FLV es el nombre con el que se conoce al contenedor de video **Flash**. En Internet, este formato comúnmente se encuentra contenido en películas Flash y es reproducido por el reproductor de Flash del navegador. Podemos obtener más información en la Guía de aprendizaje de Flash Video: **www.adobe.com/es/devnet/flash/articles/video_guide.html**.

Figura 14. Subcreator en el sitio **`http://bit.ly/qKdM3k`** es un software para trabajar con subtítulos.

La etiqueta <source>

HTML5 incorpora una etiqueta para indicar los recursos multimedia de las etiquetas de audio y video. Al escribir el código, podemos contener una o más fuentes **<source>** dentro de las etiquetas de **<audio>** o **<video>**. Esto nos posibilita indicar archivos de diferente formato para lograr compatibilidad con diversos navegadores o plataformas.

La etiqueta **<source>** soporta los siguientes atributos:

- **src**: permite indicar la URL del recurso multimedia.
- **media**: permite indicar el tipo de medio, por defecto es **all**.
- **type**: nos brinda la posibilidad de especificar el tipo de archivo o recurso multimedia. Por ejemplo, si se tratara de un archivo

SOLUCIONES MULTIMEDIA

Aunque muchos asocien el concepto multimedia al área del entretenimiento, bien vale decir que su alcance va mucho más allá. Sin ir más lejos, es una alternativa educativa muy útil, en especial si se combina con características interactivas, tanto para niños como para adultos. Esto abre una puerta muy interesante en lo que respecta a la educación presencial y, también, a distancia.

MP3, el tipo sería **audio/mpeg** o, en el caso de video WebM, correspondería **video/webm**. Además, es posible indicar el códec. Este atributo recibe información del tipo MIME.

También resulta muy importante destacar que la etiqueta **<source>** soporta los atributos globales de HTML5.

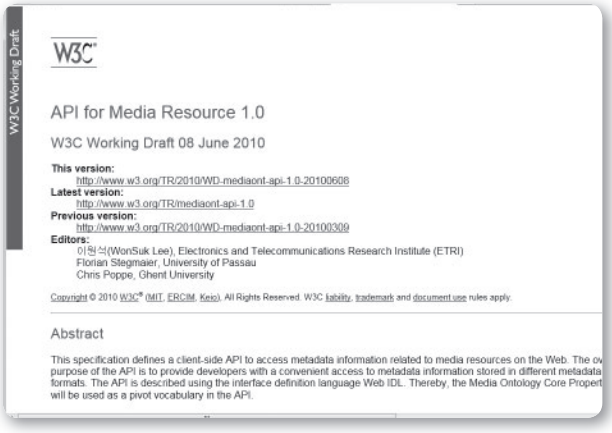


Figura 15. Para trabajar de manera avanzada con multimedia, es recomendable acceder a **www.w3.org/TR/mediaont-api-1.0**.

Aspectos de compatibilidad de audio y video

Sabemos que el tema compatibilidad ha dado mucho que hablar para el soporte multimedia de HTML5, y aún hay varias cuestiones en debate; por esta razón es un tema que debemos tener en cuenta.

En lo que se refiere a audio, el soporte del navegador Internet Explorer se introdujo a partir de la versión 9 para el formato de audio MP3. En el caso del navegador Mozilla Firefox, desde el lanzamiento de la versión 3.5 se soporta el formato Ogg Vorbis. Google Chrome, desde la versión 6, ofrece soporte tanto para el formato Ogg Vorbis como también para MP3. Pensando en el sistema operativo Mac OSX, podemos mencionar que Safari, en la versión 5, brinda soporte para MP3 y WAV. Opera, desde la versión 10.5, soporta Ogg Vorbis y WAV. Como podemos ver, el panorama se presenta variado.

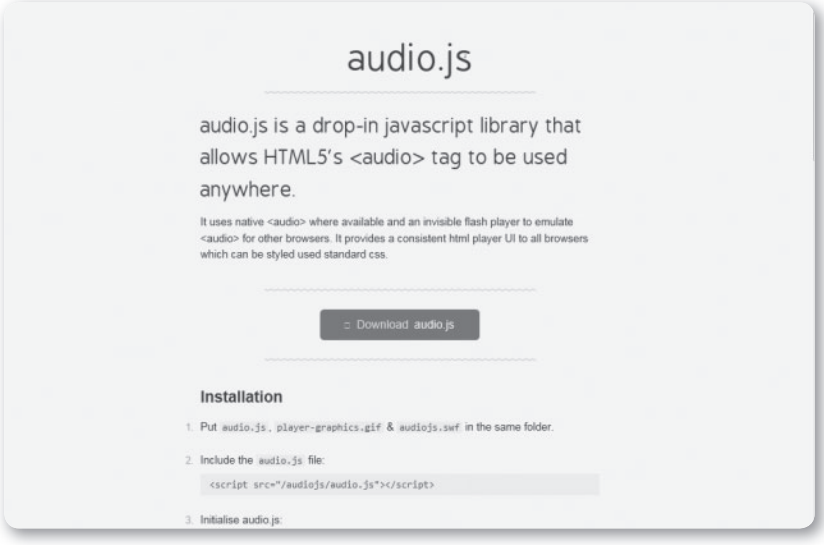


Figura 16. La librería de JavaScript **`audio.js`** (**`http://kolber.github.com/audiojs`**) permite utilizar la etiqueta de audio de HTML5 como una solución cross-browser.

En el caso de video, el navegador Internet Explorer brinda soporte a partir de la versión 9 para H.264 y VP8 (con contenedor WebM). Mozilla comenzó a soportar Ogg Theora desde la versión 3.5 de Firefox y, en la versión 4, incluyó VP8 (con contenedor WebM). Safari, desde la versión 3.1, soporta H.264. Opera, desde la versión 10.50, soporta Ogg Theora y, a partir de la 10.60, incluyó VP8 (con WebM).

Por su parte, el navegador web Chrome soporta VP8 (contenido por WebM), pero aquí vale recordar que Google realizó una importante

QUICK MEDIA CONVERTER

Si buscamos una solución potente y versátil para realizar conversiones entre diversos formatos multimedia, el programa denominado **Quick Media Converter** es una aplicación que nos brindará muy buenos resultados y gran facilidad de uso. Es freeware, y podemos obtener más información ingresando en el sitio web **www.cocoonsoftware.com**.

acción para apoyar su formato al retirar de su navegador, en febrero del año 2011, el soporte para H.264.

Ultimamente han aparecido agregados tanto para Chrome como para Mozilla Firefox, para incluir soportes adicionales a otros formatosUna herramienta interesante es www.mirovideoconverter.com.

Figura 17. **`Miro Video Converter`** es una herramienta gratuita que permite convertir video a formatos como MP4 y WebM (vp8).

Es importante aclarar que, como todavía algunos detalles se encuentran en definición, aún se pueden producir cambios en el soporte para estos formatos y códecs por parte de las nuevas versiones de los navegadores mencionados.

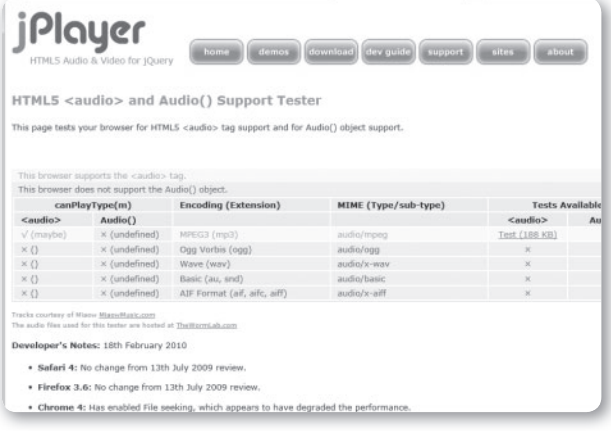


Figura 18. En **`www.jplayer.org/HTML5.Audio.Support`** hay un test de compatibilidad de las opciones de audio del navegador.

Incorporar los elementos multimedia utilizando HTML5 en nuestro desarrollo

En este apartado, analizaremos diferentes ejemplos que nos permitirán ver en acción las características de audio y video de HTML5.

En primer lugar, vamos a ver una forma sencilla de incluir audio mostrando, además, los controles correspondientes:

```
<audio src="./audio/tema01.mp3" controls>
        Texto para los navegadores no compatibles.
</audio>
```

Si deseamos especificar diferentes fuentes para ofrecer compatibilidad con diversos formatos de video, podemos hacerlo utilizando la etiqueta **<source>**, como veremos a continuación:

```
<audio controls>
  <source src="./audio/tema01.mp3" type="audio/mpeg" />
  <source src="./audio/tema01.ogg" type="audio/ogg" />
Texto para los navegadores no compatibles.
</audio>
```

Veamos ahora una porción de código muy sencillo, que nos ayudará a incorporar video en nuestro proyecto web:

```
<video src="./vid/clip01.webm" poster="./img/imgprev01.jpg" controls>
        Texto para los navegadores no compatibles.
</video>
```

En el código anterior, especificamos la ruta y el nombre del video en el ejemplo de formato WebM, una imagen para que se muestre y también que los controles estén presentes.

Tengamos en cuenta que podremos especificar diversas fuentes de la misma manera en que lo hicimos con el audio, utilizando la etiqueta **<source>** como veremos a continuación:

```
<video poster="./img/imgprev01.jpg" controls>
        <source src="./vid/clip01.mp4" type="video/mp4">
        <source src="./vid/clip01.ogv" type="video/ogg">
        Texto para los navegadores no compatibles.
</video>
```

Cuando se indica el tipo de archivo multimedia, también es posible indicar el códec, separando con punto y coma. A continuación, veamos un ejemplo de cómo quedaría la línea de la fuente dentro de la etiqueta de video para el caso de MP4:

```
<source src='./vid/clip01.mp4' type='video/mp4; codecs="avc1.42E01E,
mp4a.40.2"'>
```

Es posible que accedamos a personalizar esta configuración, con la incorporación de otras fuentes, incluyendo la opción denominada **poster** (como vimos en el listado anterior), o agregando alto y ancho (con los atributos **height** y **width**) o el arranque **automatic** (**autoplay**), entre otros atributos que vimos anteriormente.

Debemos saber que encontraremos más información relevante y también ejemplos de los tipos y códecs que podremos utilizar en el apartado **The source element** del documento de video del W3C, en el sitio web que se encuentra en la dirección **www.w3.org/TR/html5/video.html#the-source-element**.

Figura 19. La integración de videos en nuestro sitio puede realizarse tanto en el cuerpo del sitio, como en las notas o en las barras laterales.

Si deseamos comprobar que el navegador del usuario es capaz de reproducir el tipo de archivo multimedia que incluimos, ya sea audio o video, podremos utilizar el método de acceso al DOM **canPlayType**.

Para esto, en primer lugar podremos acceder a verificar si el navegador soporta audio nativo de HTML5, para lo cual utilizaremos un código JavaScript de la siguiente manera:

```
var soporteAudio = !!(document.createElement('audio').canPlayType);
```

Podríamos hacer la misma evaluación con la etiqueta de video.

El W3C nos ofrece más información sobre el uso de las etiquetas de video y ejemplos en **www.w3.org/TR/html5/video.html#dom-navigator-canplaytype**.

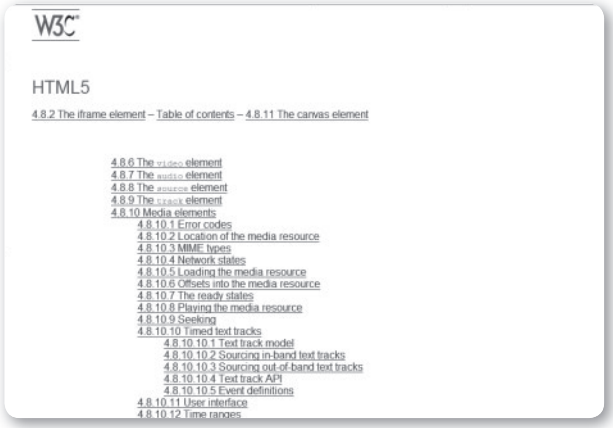


Figura 20. Encontramos la especificación de los elementos multimedia de HTML5 en **www.w3.org/TR/html5/video.html**.

En la dirección **http://dev.w3.org/html5/spec/video.html**, encontraremos la hoja de ruta W3C Editor's Draft sobre video; allí podremos leer las actualizaciones del grupo de trabajo.

Es importante señalar que, de manera predeterminada, tanto el reproductor de audio como el de video pueden tener algunas diferencias en su representación, según el navegador y el sistema del usuario. Estas diferencias en muchos casos pueden resultar sutiles; además de las opciones que vimos aquí para personalizarlos, podremos aprovechar las características que nos ofrece JavaScript. Incluso nuestra labor se verá aún más simplificada si utilizamos frameworks.

Figura 21. Si deseamos hacer podcast con WordPress podemos recurrir al sitio web **`www.blubrry.com/powerpress`**.

Potenciar y personalizar nuestro desarrollo con frameworks de JavaScript

Existen varias librerías y plugins que nos permiten mejorar las posibilidades de reproductores multimedia que tenemos por defecto, sin necesidad de programar demasiadas líneas de código.

Una alternativa es la que nos ofrece **jPlayer**. Este plugin de jQuery nos brinda una solución cross-browser para personalizar nuestro reproductor de audio o video tanto en su estilo como en los controles. Brinda soporte para HTML5 y, también, para Flash, y podemos saber más sobre este plugin si visitamos el sitio **www.jplayer.org**. Vale destacar que jPlayer es multiplataforma y funciona en toda la familia Internet Explorer (desde la versión 6) y en los navegadores Firefox, Chrome, Safari y Opera.

¿QUÉ SIGNIFICA STREAMING?

Tengamos en cuenta que las técnicas relacionadas con **streaming** en Internet permiten transmitir audio o video sin que resulte necesario que el usuario descargue todo el contenido en su disco duro. Esta alternativa brinda la posibilidad de que el usuario comience la reproducción y vaya recibiendo porciones del contenido que se van almacenando en un buffer temporal.

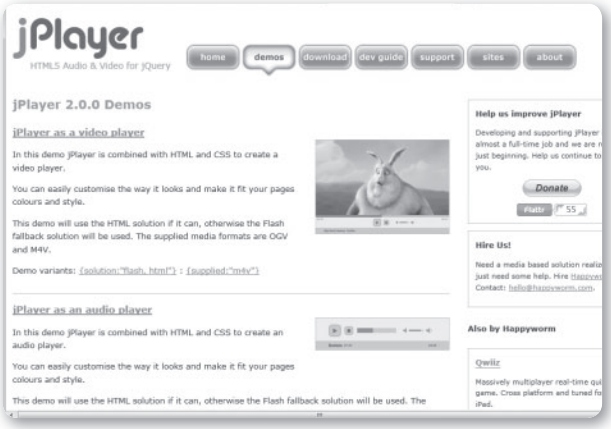


Figura 22. Si deseamos observar el reproductor en acción, podemos ver algunas demos en **`www.jplayer.org/latest/demos`**.

Otra solución interesante es la que nos ofrece **VideoJS**. Este player basado en HTML5 y JavaScript nos brinda una solución fácil de implementar, personalizable, con opciones de pantalla completa y controles. Además, proporciona interesantes ventajas al brindar un plugin para WordPress, y extensiones para Drupal y Joomla.

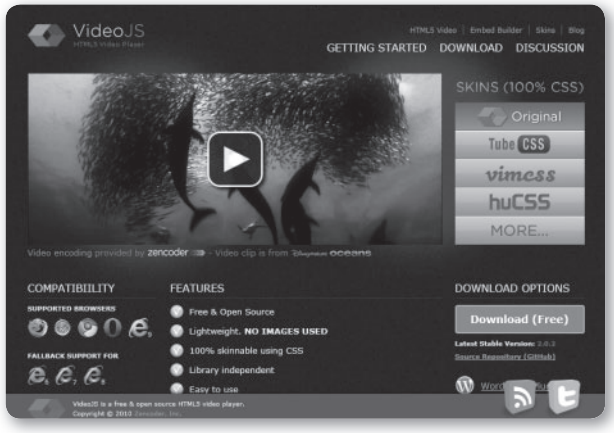


Figura 23. **`VideoJS`** se puede descargar de manera gratuita del sitio web **`http://videojs.com`**.

Debemos tener en cuenta que si elegimos la opción de descargar la última versión estable, podremos encontrar dentro del archivo comprimido en ZIP las librerías de JavaScript, las skins desarrolladas con CSS y, también, un archivo de demo en HTML.

Figura 24. Si probamos el archivo de demostración, podremos apreciar las características y la performance del producto.

En el caso de que deseemos buscar alternativas para potenciar nuestro desarrollo y nos interese contar con otro reproductor personalizable con JavaScript, podemos utilizar **MooPlay**, basado en Mootools. Cabe destacar que este reproductor, además, nos permite trabajar con subtítulos en formato SubRip (.SRT) o SubViewer (.SUB).

Figura 25. **MooPlay** se puede descargar de manera gratuita desde el sitio web **http://mooplay.challet.eu**.

Para finalizar, es importante recordar que Modernizr es una librería que nos ayudará a poder detectar las capacidades multimedia que ofrece el navegador del usuario, tanto para audio (**www.modernizr.com/docs/#audio**) como para video (**www.modernizr.com/docs/#video**).

Acceso a dispositivos multimedia

El acceso desde el navegador a cámaras web u otros dispositivos multimedia, en la actualidad, se resuelve principalmente a través de desarrollos realizados con Flash u otras plataformas que requieren de un agregado para funcionar en el navegador.

Figura 26. El acceso a dispositivos mediante Flash cuenta con configuraciones de seguridad que puede establecer el usuario.

Existen varios aspectos tecnológicos y de seguridad que se deben tener en cuenta para implementar de manera nativa el acceso a dispositivos multimedia. El W3C ha estado trabajando sobre diferentes borradores para permitir este acceso; incluso, se ha elaborado la idea de crear un elemento para dicha finalidad (**device**) o bien trabajar con las APIs que permiten transmitir flujo (**stream**).

Una de las especificaciones sobre las que se ha trabajado en el Draft es sobre **HTML Media Capture**. Esta alternativa se basa en la posibilidad de permitir el acceso a captura multimedia basada en elementos de formulario HTML, definiendo una interfaz para acceder, por ejemplo, a dispositivos como micrófono o cámara.

Esto se podría lograr por medio de los controles de formulario **<input>** que tengan **type=file**. Por ejemplo, según se muestra en el Working Draft del W3C, para subir una imagen capturada por una cámara, es posible usar el siguiente código:

```
<input type="file" accept="image/*;capture=camera">
```

Existen varias características relacionadas sobre las que está trabajando el W3C; podemos leer sobre ellas en el artículo titulado **HTML Media Capture** (**www.w3.org/TR/html-media-capture**). Estas funcionalidades aún están en desarrollo y no se encuentran extendidas en la mayoría de los navegadores actuales.

Figura 27. El acceso a dispositivos multimedia es uno de los aspectos sobre los que se está trabajando en las nuevas especificaciones.

RESUMEN

En este capítulo, hemos profundizado sobre las posibilidades multimedia que se introducen a partir de la llegada de HTML5. Analizamos en detalle las ventajas que nos brindan las etiquetas **<audio>** y **<video>**. Vimos los atributos que soportan y, luego, analizamos cómo aplicarlas en nuestros proyectos web. También conocimos el uso de las etiquetas **<source>** y <track> para agregar más fuentes y pistas de texto, respectivamente. Para finalizar, analizamos la forma en que accedemos a los dispositivos multimedia.

Actividades

TEST DE AUTOEVALUACIÓN

1. Explique qué posibilidades ofrece la etiqueta **<audio>** de HTML5.
2. Indique qué características brinda la etiqueta **<video>** de HTML5.
3. ¿Para qué sirve el atributo **loop** y cuáles de las etiquetas vistas en este capítulo lo soportan?
4. ¿Qué diferencias hay entre el soporte de audio y video que ofrece HTML5 y el que se puede lograr utilizando desarrollos basados en Flash?
5. Indique qué atributo se utiliza para indicar la URL del archivo de audio o video en las etiquetas **<audio>** y **<video>** de HTML5.
6. ¿Qué función cumple el atributo **poster** y con cuál de las etiquetas vistas en este capítulo puede trabajar?
7. ¿Es posible agregar subtítulos a un video? Indique qué opciones se pueden utilizar.
8. ¿Para qué sirve la etiqueta **<source>**?
9. ¿Qué códecs de audio y video utiliza el formato WebM?
10. Indique qué opciones se están contemplando para acceso a dispositivo multimedia.
11. actividades prácticas
12. ¿Con cuántos datos se puede llevar a cabo una comparación?

EJERCICIOS PRÁCTICOS

1. Incorpore un archivo de audio en un documento HTML4/XHTML utilizando las etiquetas **<object>** y **<embed>**.
2. Incluya un archivo de audio utilizando la etiqueta indicada en HTML5.
3. Incorpore un archivo de video utilizando la etiqueta indicada en HTML5.
4. Incluya un archivo de video que cuente con una fuente con formato WebM y otra MP4.
5. Personalice el reproductor de video utilizando uno de los frameworks de JavaScript vistos en este capítulo.

*06 -

Formularios

En este capítulo, conoceremos todos los aspectos relacionados con la creación de formularios web. Comenzaremos analizando las posibilidades que nos ofrecen HTML4 y XHTML para la creación de formularios; luego, veremos cómo se puede potenciar el desarrollo empleando JavaScript y AJAX. Para completar, profundizaremos en las características de HTML5 para la creación de formularios.

- **Evolución en el uso de los formularios en la Web 212**
- **Formularios HTML/XHTML .. 213**
  - La etiqueta <form> 213
  - La etiqueta <button> 222
- **JavaScript y AJAX para potenciar los formularios 223**
- **Características del formulario que incorpora HTML5 ..... 226**
  - Nuevos atributos para <form> 226
  - La nueva etiqueta <datalist> 226
  - La nueva etiqueta <ouput> 227
  - Las novedades en <input> 227
  - Los cambios para <textarea> 231
  - Otros cambios para etiquetas de formulario 232
- **Incorporar características HTML5 en un formulario 233**
  - Crear un formulario HTML5 desde cero 238
- **Resumen 239**
- **Actividades 240**

Evolución en el uso de los formularios en la Web

Sabemos que los **formularios** son elementos que han tenido una evolución muy importante a lo largo del tiempo en el desarrollo de sitios Web. En este sentido es interesante recordar que su misión primordial es permitir que el usuario pueda ingresar información para que sea procesada por un servidor, lugar donde se realizará una acción determinada por el código que se haya creado para dicho fin.

Tengamos en cuenta que esta operación puede ser desde el envío de un e-mail hasta la actualización de una base de datos, o también la combinación de ambas, entre muchas otras posibilidades que nos ofrecen los sitios que componen la Web actual.

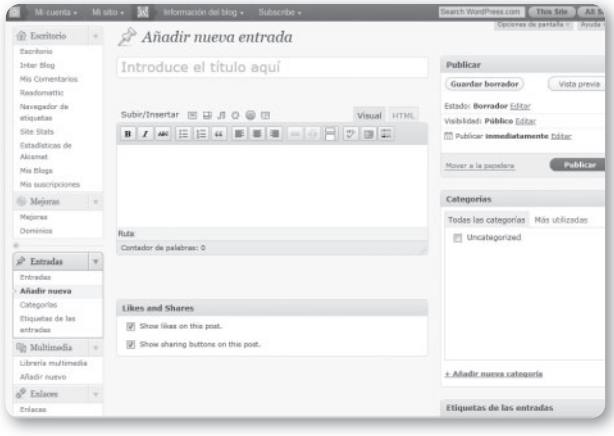


Figura 1. Un formulario puede ser parte de una aplicación mucho más compleja, como el panel de un administrador.

Los formularios se potencian con el uso de JavaScript, ya que es posible utilizar esta tecnología para validación de datos del lado cliente. Esto permite evitar el envío innecesario de los datos para que sean validados en el servidor; de esta forma se salvan transacciones innecesarias. Con los avances introducidos mediante las técnicas relacionadas con AJAX, se logró potenciar de una manera exponencial el uso de los formularios. Además, numerosos frameworks cuentan con características que facilitan la aplicación de AJAX en formularios, y simplifican notablemente la tarea del desarrollador.

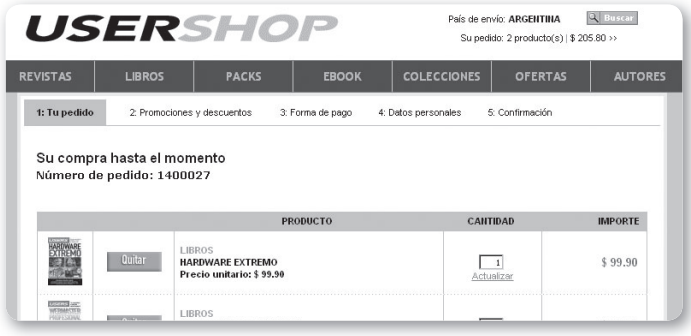


Figura 2. Los formularios resultan muy útiles para ser utilizados en aplicaciones relacionadas con el e-commerce.

Formularios HTML/XHTML

Los elementos del formulario permiten que el usuario pueda escribir o interactuar con ellos para proveer algún tipo de dato o información.

En la **Tabla 5** del **Capítulo 2**, hemos conocido las etiquetas relacionadas con formularios, para qué se utilizan y cuáles son sus principales atributos. A continuación, vamos a profundizar sobre sus características para comprender mejor su función.

La etiqueta <form>

La etiqueta denominada **<form>** es la que se encarga de contener los elementos del formulario. Sin embargo, debemos tener en cuenta que el elemento **form** no es solo un contenedor de elementos, ya que además puede recibir una serie de atributos muy importantes para definir la manera en que trabajará el formulario correspondiente.

El atributo **action** es importante ya que define la acción que realizará el formulario una vez que sea enviado. Habitualmente, recibe el valor de un recurso URL de Internet, que puede ser una página PHP o ASP que procesará el requerimiento del formulario.

Con **method**, podemos indicar el método que se utilizará para enviar el formulario. Las opciones son **get** o **post**. Con **get**, pasaremos

la información (nombre=valor) por medio de la URL y será visible en la barra del navegador. No es recomendable utilizar este método para información privada, pero puede ser útil para el caso de que se desee permitir que el usuario guarde la dirección en el historial o en favoritos. También pensemos que, al pasar la información por este recurso, contamos con un límite de caracteres, por lo cual no es conveniente cuando tenemos campos definidos con **<textarea>** o cuando se permite subir archivos. El método **post** es recomendable para las transacciones con el servidor donde no deseamos que los datos pasen por la URL; además, nos permite trabajar sin problemas con subida de archivos y campos de texto largos.

SI EN UNA PÁGINA SE UBICAN VARIOS FORMULARIOS, PODEMOS IDENTIFICARLOS CON UN ID

El atributo **accept** brinda la posibilidad de indicar el tipo MIME que puede ser subido en un campo destinado para upload (un **<input>** con **type= "file"**). Por ejemplo, se pueden indicar si son archivos de texto, imágenes, etcétera. Por compatibilidad y para mejores prestaciones, este tipo de necesidad se suele resolver utilizando técnicas AJAX.

Con **accept-charset**, es posible indicar el juego de caracteres que puede manejar el servidor que recibe la información del formulario. Por ejemplo, se pueden utilizar las opciones **UTF-8** o **ISO-8859-1**.

El atributo **enctype** nos brinda la posibilidad de indicar cómo debe ser codificada la información enviada al servidor. Puede recibir los valores **application/x-www-form-urlencoded** (valor predeterminado), **multipart/form-data** (con este valor no se codifican los caracteres, debe emplearse cuando se utiliza en el formulario algún campo con opción de upload) o **text/plain** (texto plano; en este caso, los espacios se convierten en el símbolo **+**, y el resto de los caracteres especiales se mantienen sin

USABILIDAD Y ACCESIBILIDAD

Debemos tener en cuenta que al crear un formulario, debemos tratar de que este resulte usable y accesible, ya que personas de diferentes características y con distintos tipos de dispositivos pueden utilizarlo. Algunas claves son: correcto marcado semántico, campos bien identificados, respeto del flujo de los elementos en el código y un diseño adaptable a las pantallas de diferentes dispositivos.

codificar). Para utilizar esta opción, cuando hay campos que ofrecen la posibilidad de subir archivos, debe emplearse con **method="post"**.

El atributo **target** puede recibir los valores **_blank**, **_self**, **_parent**, **_top** o el nombre de un frame. Aunque es soportado por la mayoría de los navegadores, no está recomendado su uso.

Con la etiqueta **name**, podemos proceder a indicar el nombre del formulario. Recordemos que también podemos aplicarle los atributos estándar: **id**, **class**, **dir**, **style**, **title**, **lang** y **xml:lang**.

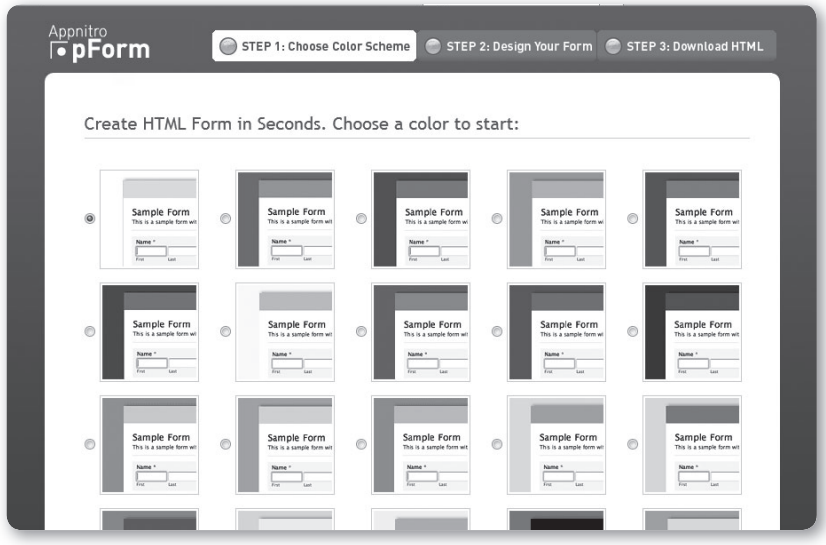


Figura 3. Si buscamos una alternativa para crear formularios de manera sencilla, podemos visitar **www.phpform.org**, y allí probar una herramienta online que nos permite crear formularios personalizados.

La etiqueta <input>

La etiqueta **<input>** nos brinda la posibilidad de definir un control de entrada en un formulario. Son varios los atributos que puede tener este elemento y, a continuación, hablaremos sobre ellos.

El atributo **accept** nos brinda la posibilidad de indicar los tipos de archivos aceptados por el campo. Este atributo de tipo MIME solo se emplea cuando el campo tiene asignado el atributo **type** con un valor **file**. Se puede indicar el tipo de archivo (documento de texto, imágenes,

etcétera), pero generalmente esto se define de una manera más eficiente y compatible utilizando técnicas AJAX. Debemos tener en cuenta que eEl atributo denominado **align** puede trabajar con campos **<input>** que tengan asignado **type="image"**. Debemos saber que puede recibir los valores **left**, **right**, **top**, **middle** o **bottom**, pero es necesario tener en cuenta que su uso está desaconsejado ya que es una necesidad que puede resolverse utilizando CSS. También, cuando el campo cuenta con **type="image"**, se puede utilizar el atributo **alt** para indicar un texto alternativo y **src** para indicar la dirección URL de la imagen.

Con **name**, podremos indicar el nombre del campo. Si deseamos indicar un valor al campo, lo hacemos con el atributo **value**.

Recordemos que si necesitamos deshabilitar un elemento determinado, debemos hacerlo con **disable**.

Con el atributo **size**, es posible indicar el ancho del elemento, aunque esta propiedad la podemos señalar con mayor precisión mediante CSS. Si necesitamos indicar un máximo de caracteres para campos de texto o contraseña, podemos hacerlo utilizando **maxlength**. Para campos de texto o que requieran una contraseña, es posible indicar si deseamos que sean de solo lectura incluyendo el atributo **readonly**.

En caso de que utilicemos **radio button** o **chekbox**, podemos indicar si deseamos que alguno esté tildado de manera predeterminada, utilizando el atributo **checked**.

El elemento **<input>** también soporta los atributos siguientes: **class**, **id**, **dir**, **lang**, **style**, **title accesskey**, **tabindex** y **accesskey**.

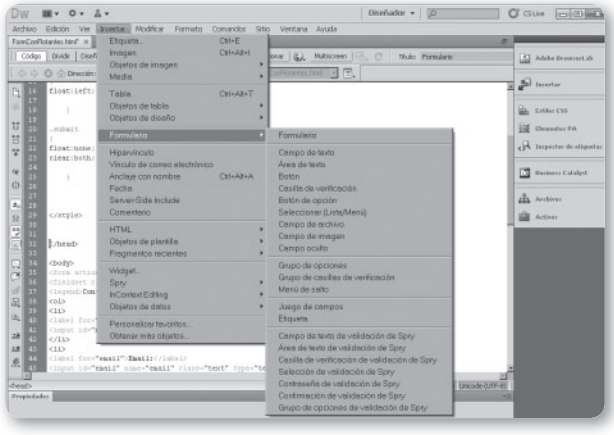


Figura 4. Dreamweaver ofrece alternativas para formularios. Podemos incorporar mediante código las opciones avanzadas.

Además de estos atributos, el elemento denominado **<input>** soporta uno muy importante que ya hemos mencionado y sobre el que profundizaremos a continuación: **type**.

Los tipos para el elemento <input>

El atributo **type** es el que nos permite definir qué tipo de control de entrada se representará y, a su vez, qué finalidad tendrá.

Los tipos que podremos definir en HTML4/XHTML son los siguientes:

- **text**: se utiliza para indicar que es un campo de texto.
- **password**: similar al campo de texto, pero destinado a contraseñas. Los caracteres que se cargan en este campo no se muestran mientras los digita el usuario.
- **file**: se emplea para definir un campo que permite subir archivos.
- **image**: permite definir una imagen como botón de envío.
- **hidden**: se utiliza para crear un campo oculto.
- **button**: permite definir un botón que formará parte del formulario (se lo puede programar mediante un script).
- **submit**: se utiliza para crear el botón de envío de un formulario.
- **reset**: se trata de una opción que nos permite crear un botón para restablecer (limpiar) los campos del formulario.
- **checkbox**: se emplea para crear un checkbox (casilla de chequeo).
- **radio**: se utiliza para crear un radio button. La diferencia entre checkbox y radio button, además de la forma cuadrada o circular que muestran, es que el primero permite seleccionar varias opciones y el segundo, solo una.

HOT SCRIPTS

El sitio llamado **Hot Scripts** se presenta como una gran fuente de recursos relevantes para todos aquellos que realizan desarrollo de sitios web. Allí es posible encontrar una importante variedad de scripts de diferentes lenguajes y tecnologías, dentro de las que podemos destacar HTML5, aunque también encontraremos recursos relacionados con PHP, JavaScript, AJAX, Ruby on Rails, ASP, ASP.NET, CGI y Perl; todo lo que ofrece este sitio se encuentra acompañado de un diseño atractivo. Debemos recordar que el sitio web de Hot Scripts se ubica en la dirección **www.hotscripts.com**.

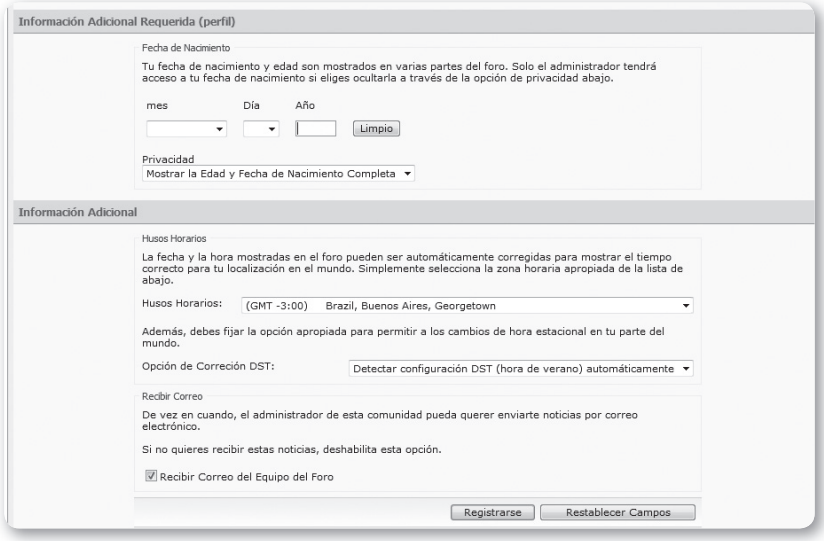


Figura 5. Un formulario puede estar compuesto por texto y por diferentes tipos de controles de entrada; esto le ofrece, al usuario, diversas posibilidades de ingresar los datos.

En HTML5, como podremos ver más adelante, se incorporan nuevos tipos para el elemento **<input>.**

La etiqueta <label>

El elemento denominado **label** nos permite incluir un texto para un campo de formulario. En palabras sencillas, se trata del texto visible que acompaña, por ejemplo, a un **<input>**.

Para relacionar la etiqueta **<label>** con el control de entrada correspondiente, se utiliza el atributo **for**. El atributo **accesskey** nos brinda una tecla de acceso rápido al elemento. Estas características, sumados a los atributos estándar, nos permiten lograr un formulario accesible y bien estructurado. Veamos a continuación un ejemplo:

```
<label for="print_model" accesskey="p">Modelo de impresora</label>
<input type="text" name="modelo" id=" print_model ">
```

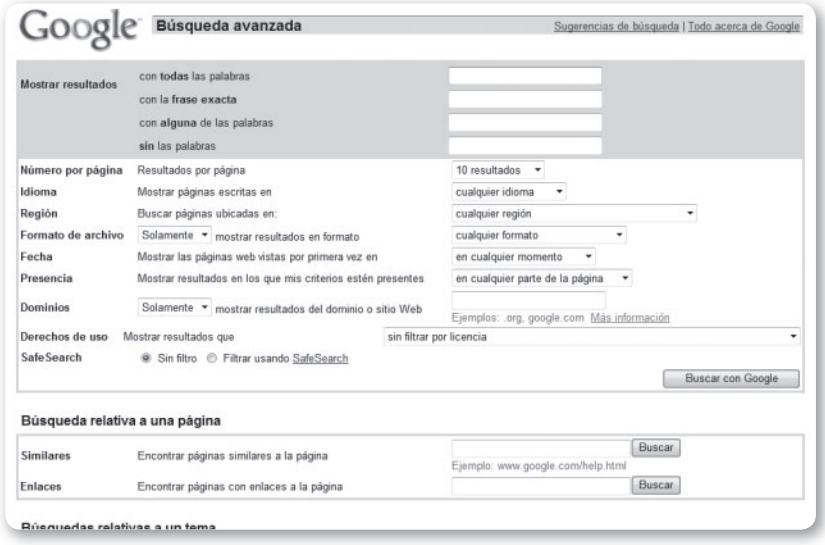


Figura 6. Uno de los usos que se les puede dar a los formularios es el de creación de cuadros de búsqueda, como por ejemplo el que nos ofrece el **Buscador Avanzado de Google**.

La etiqueta <textarea>

Con la etiqueta **<textarea>**, es posible realizar la definición de un control de entrada de múltiples líneas. Para este elemento, podemos indicar el número de filas y columnas visibles, asignando un valor numérico a los atributos **rows** y **cols**, respectivamente.

Además de los atributos estándar, contamos con atributos para deshabilitar este control (**disable**), hacerlo de solo lectura (**readonly**), asignarle una tecla de acceso rápido (**accesskey**), indicar un orden

PHP CLASSES

Si trabajamos con PHP, con seguridad nos será muy útil poder conseguir clases que nos ayuden a resolver diferentes necesidades de desarrollo. **PHP Classes** es un sitio que nos brinda acceso a distintos tipos de scripts de clases de PHP, tutoriales, guías, foros y muchos otros recursos e información interesante. La dirección del sitio web para acceder es **www.phpclasses.org**.

(**tabindex**) y aplicarle un nombre (**name**). A continuación, veremos un ejemplo de un área de texto de 5 filas y 25 columnas.

```
<textarea rows="5" cols="25"></textarea>
```

Figura 7. `Email Me Form` (`www.emailmeform.com`) es una herramienta online que nos permite crear un formulario en pocos pasos.

Las etiquetas <fieldset> y <legend>

Con la etiqueta **<fieldset>**, podremos agrupar un conjunto de elementos dentro de un formulario. Esto puede ser muy útil para agrupar elementos que formen una sección dentro de él.

Visualmente, de manera predeterminada, este elemento dibuja una caja en torno a los elementos que agrupa.

Junto a esta etiqueta puede trabajar **<legend>**, que permite especificar un texto o una leyenda para el conjunto.

¿CLIENTE O SERVIDOR?

Debemos tener en cuenta que, para evitar que se realicen peticiones innecesarias, como alternativa de desarrollo, es muy recomendable hacer desde el lado cliente todo lo que se pueda validar eficientemente desde allí, sin que represente riesgos de seguridad. Algunas validaciones que requieran procesos más complejos o la realización de consultas a bases de datos pueden realizarse utilizando programación del lado servidor; de esta forma estaremos seguros de haber tomado la mejor decisión.

Ambas trabajan con los atributos estándar y pueden ser personalizadas con CSS. En el caso de **<legend>**, veremos que soporta además **accesskey** y **align**, pero, desde hace tiempo, el uso de este último está desaconsejado, ya que su función se puede resolver mediante CSS.

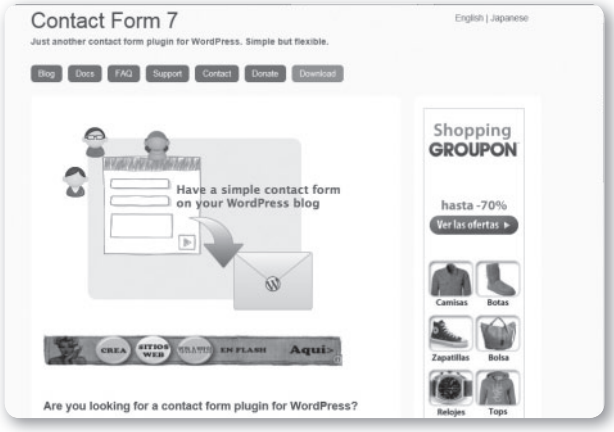


Figura 8. Para WordPress, existen plugins para crear formularios, como **`Contact Form 7`** (en **`http://contactform7.com`**).

Las etiquetas <select>, <option> y <optgroup>

Con **<select>**, podemos definir una lista desplegable dentro de un menú. Cada ítem seleccionable de esta lista desplegable se puede definir mediante **<option>**. Con **<optgroup>**, podemos agrupar elementos de una lista que sean afines. Veamos un ejemplo a continuación:

```
<select>
  <optgroup label="Monitores">
    <option value="crt">CRT</option>
    <option value="lcd">LCD</option>
  </optgroup>
  <optgroup label="Impresoras">
    <option value="laser">Laser</option>
    <option value="deskjet">Deskjet </option>
  </optgroup>
</select>
```

En cuanto a los atributos, además de los estándar, la etiqueta **<option>** puede recibir **label** (texto corto que define a la opción), **disabled** (si deseamos deshabilitarla), **selected** (si deseamos que aparezca seleccionada) y **value** (valor que se envía mediante el formulario).

Por su parte, **<optgroup>** necesita contar con un texto (**label**) que permite especificar una descripción de los elementos de opción agrupados. Si lo deseamos, podemos deshabilitar con **disabled**.

Figura 9. En esta página, podemos ver integrado el ejemplo de un `<select>` con diferentes opciones y agrupamientos.

La etiqueta <button>

Existe la posibilidad de definir un botón en un formulario mediante el elemento **<button>**, pero recordemos también que, mediante el elemento **<input>**, podemos definir un tipo que sea botón (**type="button"**).

En el caso de la etiqueta **<button>**, además de los estándar, contamos con los atributos **disabled** (deshabilitado), **name** (nombre del botón) y **value** (valor del botón que podrá ser enviado por el formulario). También podemos utilizar el atributo **type** para definir el tipo de botón, asignando los valores **button**, **reset** o **submit**, dependiendo de lo que corresponda.

JavaScript y AJAX para potenciar los formularios

El uso de JavaScript ofrece interesantes opciones para validar formularios en el equipo local. Con el uso de AJAX, se pueden potenciar estas posibilidades y, además, incorporar nuevas características para darle mayor dinamismo en el refresco asincrónico que permiten las técnicas asociadas.

Figura 10. La opción de recargar solo una parte de la página se puede ver, por ejemplo, en las encuestas que suelen colocarse en las sidebar de algunos sitios web y también en diversos blogs.

Sería demasiado extenso analizar todas las posibilidades que nos brinda AJAX para formularios, y es una materia que está más allá del alcance planteado para este libro, por lo tanto no profundizaremos en el tema. Por esta razón, en este apartado veremos algunas soluciones prácticas basadas en frameworks de JavaScript.

Para validación y manejo de formularios en general, jQuery nos ofrece una buena base de posibilidades.

jQuery plugin validation es una librería que nos facilitará el trabajo; podemos obtenerla de manera gratuita ingresando en **http://bassistance.de/jquery-plugins/jquery-plugin-validation**. Más adelante, hablaremos sobre esta librería y, también, la veremos en acción, ya que la utilizaremos para el ejemplo práctico que analizaremos para cerrar el capítulo.

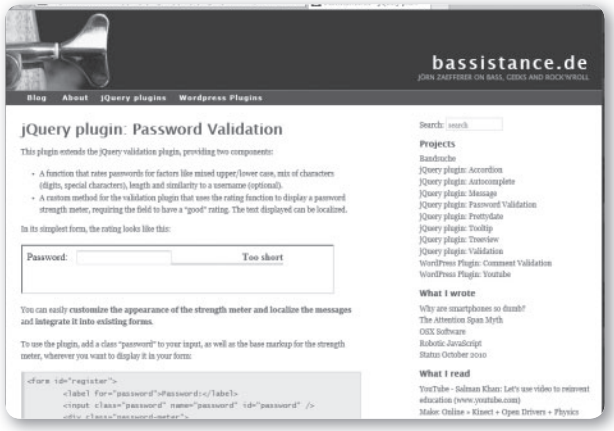


Figura 11. Si queremos validar contraseñas, podemos probar las opciones que nos entrega **`jQuery validation plugin`**.

Otras alternativas de validación interesantes son:

- **FormCheck**: una clase de Mootools desarrollada para validación, disponible en **http://mootools.floor.ch/en**.
- **jQuery Ketchup Plugin**: este plugin nos ofrece originales mensajes en forma de globos de diálogo al validar. Está disponible en **https://github.com/mustardamus/ketchup-plugin**.
- **JSValidate**: liviano y fácil de utilizar, se puede obtener en **www.jsvalidate.com**, donde también se accede a la documentación.

JQUERY UI PARA AUTOCOMPLETAR

Recordemos que si necesitamos incorporar campos en nuestro formulario que tengan ayuda para autocompletado de datos, encontraremos una solución muy interesante con **jQuery UI**. En el sitio web que se encuentra en la dirección **http://jqueryui.com/demos/autocomplete**, hallaremos muestras de código, ejemplos para probar y más información relacionada con el tema

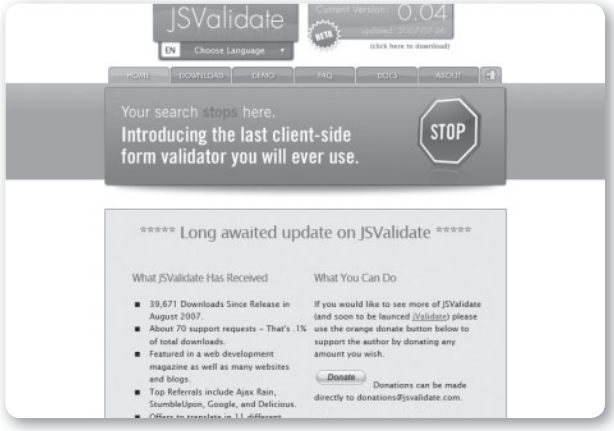


Figura 12. La librería **`JSValidate`** es una opción muy fácil de utilizar para validar formularios. Requiere Scriptaculous y Prototype.

Captcha es el nombre con el que se conoce a una técnica que se aplica para determinar si quien está ingresando los datos es una persona o, por el contrario, un agente informático. Se incluye en los formularios como una alternativa de seguridad para evitar que sean completados de manera automática. Puede ser útil en algunos casos para evitar spam u otras técnicas no deseadas.

Figura 13. **`reCAPTCHA`** es uno de los servicios de Captcha más populares. Podremos saber más en **`www.google.com/recaptcha`**.

Para finalizar, también relacionado con frameworks de AJAX, una solución de desarrollo que nos puede resultar útil en algunos casos es mostrar el formulario en una ventana modal.

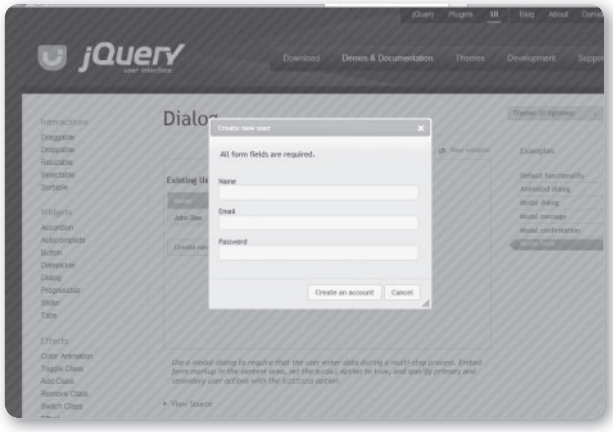


Figura 14. **`jQuery UI`** nos permite mostrar un formulario en ventana modal: **`http://bit.ly/oQgZBl`**.

Características del formulario que incorpora HTML5

Los formularios vivieron su última gran revolución con las ventajas que introdujo AJAX. Existen numerosos frameworks que nos permiten potenciar las posibilidades que nos ofrecen estos elementos.

HTML5 plantea ir un paso más allá con nuevas características que modernizan el lenguaje y brindan ventajas que podemos aprovechar.

Nuevos atributos para <form>

Para la etiqueta **<form>** en HTML5, se agrega el atributo **novalidate**, que permite indicar que el formulario no sea validado antes de ser enviado.

También, se incorpora **autocomplete** para indicar si el formulario puede ser autocompletado. El atributo **accept** deja de ser soportado.

La nueva etiqueta <datalist>

El elemento **<datalist>** se incorpora a partir de HTML5 para trabajar en conjunto con **<input>** y definir una lista de autocompletado. En la construcción de la lista, podemos usar **<option>** para definir cada ítem.

La etiqueta **<datalist>** soporta los atributos globales de HTML5 y, a continuación, veremos un ejemplo de su uso:

```
<input list="hardware" />
<datalist id="hardware" value="Elija el hardware">
        <option value="monitor">
        <option value="impresora">
        <option value="teclado">
        <option value="mouse">
        <option value="scaner">
</datalist>
```

La nueva etiqueta <ouput>

El elemento **output** se incorpora en HTML5 para permitir la representación del resultado de un cálculo en un formulario.

Este campo podría ser útil para mostrar los valores que se obtengan en cálculos matemáticos que se realicen mediante algún script.

Además de los atributos globales de HTML5, también soporta **for** (puede recibir la **id** del elemento o elementos a los que está relacionado), **form** (puede recibir la **id** del formulario al que pertenece) y **name** (es el **name** del elemento).

La etiqueta **<output>** puede trabajar con los eventos de ventana, teclado, mouse, media y formulario. Dentro de estos últimos, puede ser útil trabajar en un script con **onforminput** para realizar los cálculos con los valores que tomen los controles de entrada que vinculemos con el output que tendrá el resultado.

Las novedades en <input>

Entre las características más importantes que HTML5 incorpora para formularios, se destaca lo relacionado con los controles de entrada.

Para profundizar en este tema comenzaremos con los atributos que incorpora el control denominado **<input>** con HTML5:

- **autocomplete**: permite indicar si el campo podrá tener la característica de autocompletado (valor **on**) o no (valor **off**).
- **autofocus**: el campo que reciba este atributo tendrá el foco al ser cargado el formulario con el cual estamos trabajando.

- **placeholder**: nos permite especificar un texto que será el que se muestra en el campo del formulario cuando no está enfocado y el usuario todavía no escribió sobre dicho control.
- **form**: se emplea para indicar a qué formulario pertenece el campo. Recibe como valor el nombre de dicho formulario.
- **formaction**: este atributo está destinado a un **<input>** con **type** igual a **submit** o **image**. Este atributo permite indicar una URL para una acción. Entra en acción cuando el formulario es enviado utilizando dicho control como método de envío.
- **formmethod**: se trata de un atributo que podemos utilizar en los **<input>** con **type** igual a **submit** o **image**. Con este atributo es posible indicar el método con el que el formulario enviará la información al utilizar este control. Puede tomar los valores **get** o **post**.
- **formenctype**: también está dirigido a un **<input>** con **type** cuyo valor sea igual a **submit** o también a **image**. Permite indicar un **enctype** que tendrá preponderancia si se utiliza este control para el envío. Puede recibir alguno de los siguientes valores: **application/x-www-form-urlencoded**, **multipart/form-data** o **text/plain**.
- **formtarget**: se utiliza para sustituir el atributo **target** del formulario. Permite indicar la ventana destino del formulario cuando sea enviado. Es posible utilizarlo cuando el **<input>** tiene el valor **type="submit"** o **type="image"**. Puede tomar los valores **_blank**, **_self**, **_parent** o **_top**.
- **formnovalidate**: este atributo puede utilizarse para indicar que un campo no sea validado cuando se envíe el formulario.
- **height**: es un atributo que nos permite indicar el alto (porcentual o en píxeles) de un **<input>** cuando el **type** es **image**.
- **width**: se emplea para indicar el ancho (porcentual o en píxeles) de un **<input>**. Se puede utilizar sólo cuando type tiene el valor **image**.
- **list**: se trata de un atributo que puede recibir el **id** de un **datalist** que se relacione a ese control de entrada.
- **mix**: posibilita que se indique el mínimo valor (numérico o de fecha) que se puede ingresar en el campo.
- **max**: permite que se especifique el máximo valor (numérico o de fecha) que puede ingresarse en el campo.
- **step**: este interesante atributo nos permite recibir un número que indica los intervalos o pasos de ese campo.
- **multiple**: es un atributo que podemos emplear para permitirle al usuario que ingrese más de un valor en el campo.

- **pattern**: se trata de un interesante atributo que nos permite especificar un patrón para el campo correspondiente.
- **required**: se utiliza para indicar que el campo debe ser completado antes de que el formulario sea enviado.

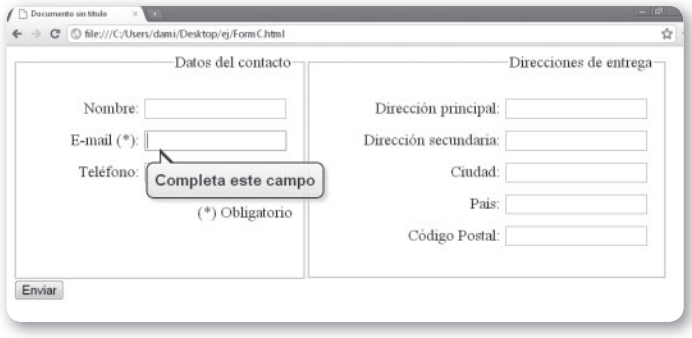


Figura 15. El atributo **`required`** ofrece la opción de validación nativa del lenguaje para los campos de entrada del formulario.

El atributo denominado **align** deja de ser soportado por el elemento **<input>** a partir de HTML5. Como veremos a continuación, para el atributo **type** se incorporan nuevos valores posibles que potencian en gran medida las opciones para **<input>**.

Los nuevos tipos para <input>

El atributo **type** del elemento **<input>** recibe una importante renovación en HTML5, que amplía en gran medida el espectro de posibilidades que puede brindar.

Por un lado, mantiene los valores que conocimos en HTML4, ellos son: **button**, **checkbox**, **file**, **hidden**, **image**, **password**, **radio**, **reset** y **text**. Pero lo realmente importante está en los nuevos valores que se incorporan a partir de HTML5. Entre ellos encontramos:

- **color**: permite crear un campo en el cual se desplegará una paleta para que el usuario pueda elegir un color.
- **date**: con este valor, es posible crear un campo que desplegará un calendario en el cual se podrá seleccionar una fecha, de la cual se podrá obtener el año, el mes y el día.

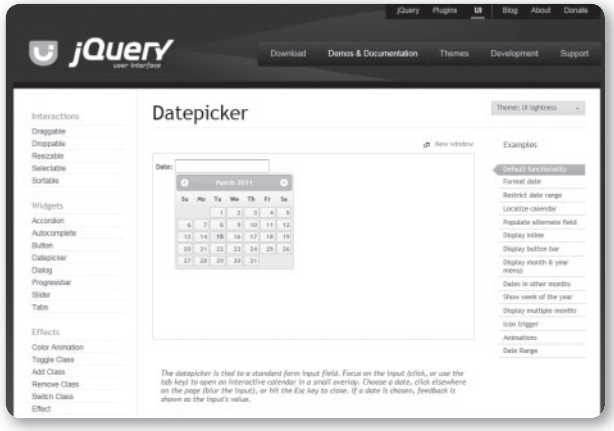


Figura 16. Para seleccionar fechas, una solución cross-browser alternativa **`http://jqueryui.com/demos/datepicker`**.

- **datetime**: permite desplegar un calendario (para seleccionar año, mes y día), y además elegir hora y minuto.
- **datetime-local**: similar al anterior, pero para la hora local.
- **month**: este elemento nos brinda la posibilidad de desplegar un calendario para elegir año y mes.
- **week**: permite desplegar un calendario para elegir una semana (el número de semana de un determinado año).
- **time**: debemos tener en cuenta que este elemento se puede utilizar para seleccionar minuto y segundo.
- **email**: se emplea en los campos destinados a correo electrónico.
- **number**: permite crear un campo de entrada para números. En este tipo, se pueden utilizar los atributos que vimos anteriormente para mínimo (**min**), máximo (**max**) y pasos (**step**).

¿USAR HTML5 EN FORMULARIOS?

Es importante tener en cuenta que el lenguaje HTML5 incluye una serie de opciones muy interesantes para potenciar el desarrollo de formularios; de esta forma, se convierte en el compañero ideal para el desarrollador. Lo que debemos recordar a la hora de utilizar cada una de estas ventajas es la necesidad del desarrollo y la compatibilidad que ofrecen. Por esta razón, vale la pena evaluar en que caso resulta conveniente emplear cada una de estas características. Por lo general, podremos utilizar técnicas AJAX que nos permitan subsanar este último aspecto, así no será necesario dejar de lado esta interesante alternativa.

Figura 17. Los campos numéricos y los de color tienen muchas aplicaciones y son una mejora importante para diversos tipos de formularios en sitios web.

- **range**: este tipo nos permite crear un control de entrada que mostrará un deslizador (slider) para seleccionar un valor numérico. Al igual que **number**, puede trabajar con los atributos **min**, **max** y **step**, y además puede tener un valor prefijado con **value**.
- **search**: es un tipo de campo que puede resultarnos de utilidad para incluir en nuestro proyecto una caja de búsqueda.
- **tel**: campo destinado para números de teléfono.
- **url**: este tipo de campo es útil para direcciones URL.

Vale destacar que, en los navegadores no compatibles con alguno de los tipos que se agregan en HTML5, se mostrará un campo de texto.

Los cambios para <textarea>

Además de los atributos propios que hemos visto para **<textarea>** en HTML4 (**cols**, **disabled**, **name** y **readonly**), a partir de HTML5 se incorporan algunos de los que hemos visto ya para el elemento **<input>**; ellos son: **autofocus**, **placeholder**, **form** y **required**.

Además, se agregan los siguientes:

- **dirname**: puede recibir el nombre del campo que contiene el texto de la dirección del área de texto.
- **maxlength**: puede recibir como valor un número que indique el máximo de caracteres para esta área de texto.
- **wrap**: se emplea para indicar la manera en que el texto será envuelto en el área de texto. Puede recibir los valores **hard** o **soft**.

Otros cambios para etiquetas de formulario

Dentro de las etiquetas que representan a los elementos de formulario, encontramos cambios relacionados con los atributos que soportan. A continuación analizaremos las principales modificaciones:

- **<label>**: esta etiqueta, además del atributo **for**, ahora también puede recibir el atributo denominados **form**.
- **<select>**: además de **disabled**, **multiple**, **name** y **size**, con la llegada de HTML5 se agregan **autofocus** y **form**.
- **<button>**: además de **name**, **type** y **value**, en HTML5 se suman los atributos que conocimos para **<input>**; ellos son **autofocus**, **form**, **formaction**, **formenctype**, **formmethod**, **formnovalidate** y **formtarget**.

Vale remarcar que **<option>** no tiene cambios en los atributos que soporta, pero la modificación se produce en que ahora esta etiqueta, además de trabajar con **<select>**, también puede hacerlo con **<datalist>**.

Por su parte, la etiqueta denominada **<optgroup>** no tiene cambios en sus atributos con la llegada de HTML5.

En el caso de **<legend>**, puede soportar los atributos globales de HTML5, pero deja de soportar el atributo **align**, ya que esta característica

EL FORMULARIO DE CONTACTO

Al definir un **formulario de contacto**, es importante pedir de manera obligatoria solo los datos imprescindibles, para evitar que el usuario desista en completarlo. Además, debemos pensar en la otra parte del desarrollo, cómo lo procesaremos del lado servidor, si enviaremos un e-mail o si agregaremos la información en una base de datos, en cuyo caso deberemos planear su estructura.

debe ser establecida mediante el uso de CSS. Otro cambio que recibe es que ahora no es de uso exclusivo para el elemento de formulario **<fieldset>**, ya que también puede trabajar con **<figure>** y **<details>**.

Incorporar características HTML5 en un formulario

Ahora que ya hemos conocido las diversas características de los formularios en HTML, vimos cómo podemos potenciarlos con AJAX y aprendimos las ventajas introducidas por HTML5, es un buen momento para poner manos a la obra.

El formulario de contacto es uno de los de uso más frecuente en el desarrollo web; por esta razón, lo tomaremos como sujeto de estudio para este ejemplo, aunque lo que veamos nos servirá de base para cualquier otra necesidad de desarrollo.

Comenzaremos con una base de un formulario en HTML4/XHTML, por ejemplo, un formulario que podríamos tener anteriormente. En el siguiente paso, le aplicaremos una validación con AJAX, empleando un plugin de jQuery para realizar un proceso más sencillo. Para finalizar y mejorarlo, le agregaremos características HTML5.

En este caso, arrancaremos con un formulario de contacto que le requiera al usuario los siguientes datos en forma obligatoria: nombre y apellido, dirección de e-mail y también un comentario. Como dato adicional no obligatorio, se le solicitará la dirección de su sitio web. Esta necesidad se puede resolver con un formulario muy sencillo como el que veremos en el siguiente listado:

PROTOTYPE

El popular framework de JavaScript denominado **Prototype**, entre las múltiples opciones que se encarga de ofrecernos, nos brinda también la posibilidad de trabajar con formularios. Dar foco, habilitar, deshabilitar, obtener el valor, seleccionar o serializar son algunas de las interesantes opciones que nos ofrece esta librería para facilitar nuestros desarrollos con formularios.

```
<form id="FormContacto" method="post" action="url">
        <label>Nombre y Apellido</label>
    <input name="nombre" type="text" id="camp_nombre" />
    <label>E-mail</label>
        <input name="email" type="text" id="campo_email"/>
    <label>Sitio web</label>
        <input name="website" type="text" id="campo_website"/>
    <label>Comentario</label>
    <textarea name="comentario" id="campo_comentario"></textarea>
    <input type="submit" name="button" id="enviar" value="Enviar" />
</form>
```

Hemos resuelto los campos de entrada mediante **<input>** del tipo texto, salvo el caso del comentario, que es un **<textarea>**. Las etiquetas son **<label>**, y el botón de envío es un **<button>** con el valor **submit**. Para este formulario, elegimos el método **post**, ya que se enviará un campo largo como el caso del comentario, que es un **<textarea>**; además, es importante que recordemos colocar, como valor del atributo **action**, la dirección URL del recurso o archivo del servidor que procesará el formulario.

Para darle estilo y ubicar los elementos de acuerdo con el diseño, podemos utilizar lo visto en el **Capítulo 4**, cuando aprendimos a utilizar CSS. Luego de definidos los estilos, pasaremos a la validación.

Emplearemos un plugin de jQuery para agregarles una validación a los campos obligatorios del formulario. Primero descargamos el plugin en **http://bassistance.de/jquery-plugins/jquery-plugin-validation**; allí podremos leer también sobre su licencia y modo de uso. En el empaquetado, podremos ver librerías y ejemplos aplicados. Recordemos que jQuery se puede obtener ingresando en **http://jquery.com**.

Colocamos las librerías JS necesarias en una carpeta de nuestro proyecto y hacemos referencia a los archivos desde la página donde se encuentre incluido el formulario que creamos anteriormente.

Veamos, a continuación, el ejemplo en el que utilizamos los archivos "min" de jQuery y del plugin de validación (podemos incluirlos en la cabecera del archivo HTML que contiene el formulario):

```
<script type='text/javascript' src='./js/jquery.js'></script>
<script type="text/javascript" src="./js/jquery.validate.js"></script>
```

Es importante indicar correctamente en el atributo **src** la ruta y el nombre del archivo. Debemos saber que la versión de jQuery puede variar, pero siempre debemos asegurarnos de que sea compatible con la versión de plugin que estamos empleando.

Debajo de los scripts incluidos, debemos inicializar el plugin, como veremos en el código que se muestra a continuación:

```
<script type="text/javascript">
$(document).ready(function() {
        $("#FormContacto").validate();
});
</script>
```

En el listado anterior, es necesario indicar el nombre de la **id** del formulario, en este caso, es **FormContacto**. En el ejemplo, lo indicamos en la línea **$("#FormContacto").validate();**.

A continuación, debemos colocarles una clase a los campos que deseamos validar. Si solo necesitamos que sea completado por el usuario, podemos utilizar la clase definida en el plugin que se denomina **required**. Esto deberíamos aplicarlo a los campos definidos para nombre y apellido, e-mail y también para el área de texto (el sitio web no es obligatorio). Veamos cómo queda el campo de nombre y apellido:

```
<input name="nombre" type="text" id="campo_nombre" class="required" />
```

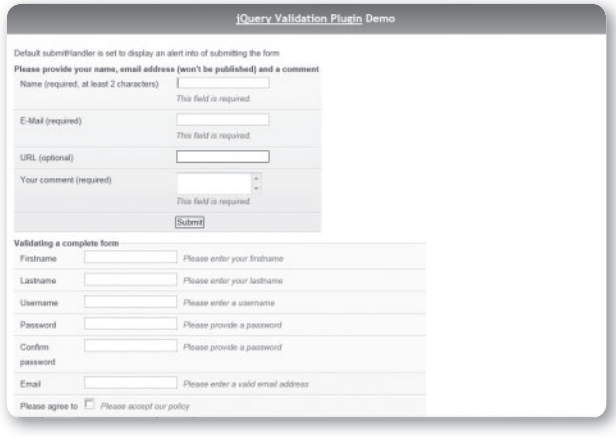


Figura 18. Los mensajes de error de **jQuery Validation Plugin** están en inglés, pero es posible personalizarlos en español.

Si queremos tener una validación específica para el campo de e-mail, le debemos agregar una clase más denominada **email**. Veamos entonces cómo quedaría esa línea:

```
<input name="email" type="text" id="campo_email" class="required email"/>
```

Debemos saber que otro de los aspectos que necesitamos tener en cuenta y que resulta muy importante es que precisamos revisar los archivos CSS que se incluyen junto al plugin de validación para integrarlos a nuestro diseño. Si no lo hacemos, es posible que la validación se visualice de manera incorrecta.

Resuelto esto, ahora nos encargaremos de pasar a realizar la aplicación de las características de HTML5. En este ejemplo, podríamos darle el foco al primer campo del formulario, establecer un texto temporal para indicarle al usuario qué debe completar en los campos y, además, establecer los campos específicos que nos provee HTML5 para direcciones de e-mail y sitios web.

El código con todos los cambios aplicados es el siguiente:

```
<form  id="FormContacto" method="post" action="url">
<label>Nombre y Apellido (dato obligatorio)</label>
    <input name="nombre" type="text" id="campo_nombre" class="required"
autofocus placeholder="Escribe tu nombre y apellido" />
    <label>E-mail (dato obligatorio)</label>
<input name="email" type="email" id="campo_email" class="required email"
placeholder="Escribe tu dirección de correo" />
    <label>Sitio web</label>
<input name="website" type="url" id="campo_website" placeholder="Escribe
la dirección de tu sitio web" />
    <label>Comentario (dato obligatorio)</label>
    <textarea name="comentario" id="campo_comentario" class="required"
placeholder="Escribe un comentario para enviar"></textarea>
    <input type="submit" name="button" id="enviar" value="Enviar" />
</form>
```

Luego de analizar la información que comentamos hasta aquí, y ver el código que acabamos de presentar, podemos observar el resultado en la imagen que mostramos a continuación.

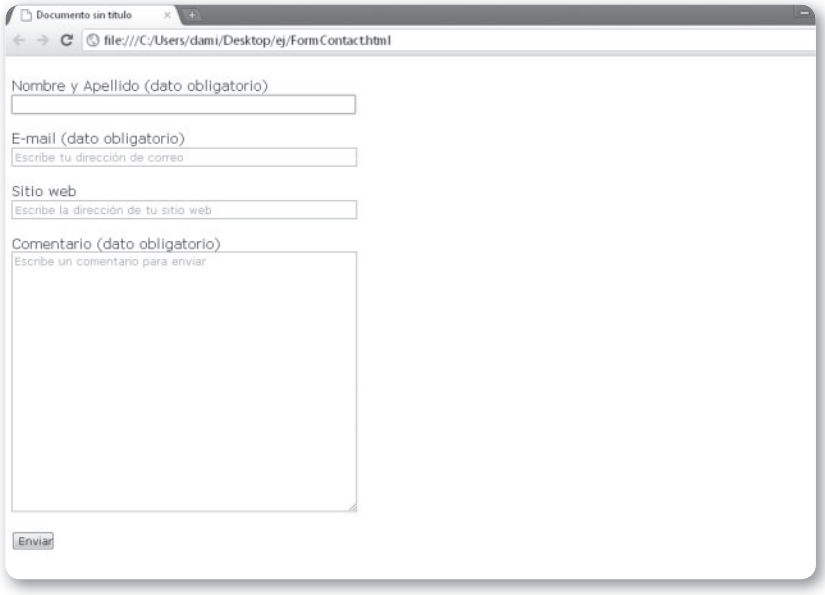


Figura 19. Si aplicamos los estilos y previsualizamos el código en un navegador compatible con HTML5, podremos ver el resultado.

Debemos tener en cuenta que es muy importante que también tengamos en cuenta el resto de las posibilidades que nos ofrece HTML5 para incluir otras características específicas al formulario, como pueden ser las opciones de validación nativas del lenguaje con el atributo **required**. Para continuar, veamos cómo quedaría el campo de entrada para nombre y apellido, con este atributo aplicado:

SERVIDORES WAMP Y LAMP

Tengamos presente que si deseamos probar el procesamiento de datos del lado servidor, podemos hacerlo instalado un **WAMP** (se trata de un servidor Apache para Windows con MySQL y PHP incluidos) o también un **LAMP** (un servidor Apache para Linux con MySQL y PHP incluidos) en nuestro propio equipo. **XAMPP** (se encuentra en el sitio web que está en la dirección **www.apachefriends.org/es/xampp.html**) es una alternativa con soporte tanto para Apache como para MySQL, PHP y Perl, que ofrece soluciones para diversas plataformas.

```
<input name="nombre" type="text" id="campo_nombre"  required autofocus
placeholder="Escribe tu nombre y apellido" />
```

También nos puede resultar útil utilizar el atributo **pattern** para efectuar una validación donde realizaremos una comparación frente a un patrón. Veamos un esquema básico aplicado al campo de e-mail:

```
<input name="email" type="email" id="campo_email"  required pattern=
"[^ @]*@[^ @]*" placeholder="Escribe tu dirección de correo" />
```

Crear un formulario HTML5 desde cero

En el ejemplo anterior, tomamos como base el inicio de un formulario escrito en HTML4/XHTML. Si deseamos desarrollar un formulario desde cero, podemos aplicar las características de HTML5 al proyecto.

Siempre basándonos en las necesidades del ejemplo anterior, al realizar el primer paso, podríamos definir la estructura HTML del documento mediante la incorporación allí mismo de las características de HTML5. Esto consiste en establecer los atributos **placeholder** y **autofocus**. Además, podríamos definir el **type** correspondiente a cada campo en el caso particular de los destinados a correo electrónico y a sitio web. Esto nos permitirá ahorrar el paso final, ya que tendremos definidas estas características desde el principio.

WAMPSERVER

WampServer es una solución que nos puede resultar muy útil si trabajamos con Windows y necesitamos montar un servidor Apache para probar desarrollos que requieran el uso de código PHP y bases de datos MySQL. El paquete también incluye PhpMyadmin y SQLBuddy. El archivo de instalación se encuentra disponible en el sitio web **www.wampserver.com/en**. WampServer se ofrece dentro del marco de un proyecto del tipo open source y su licencia de uso es GPL. Cuenta con versiones para sistemas Windows de 32 y de 64 bits. Una posibilidad interesante es la capacidad de extender las funcionalidades de los módulos mediante Addons.

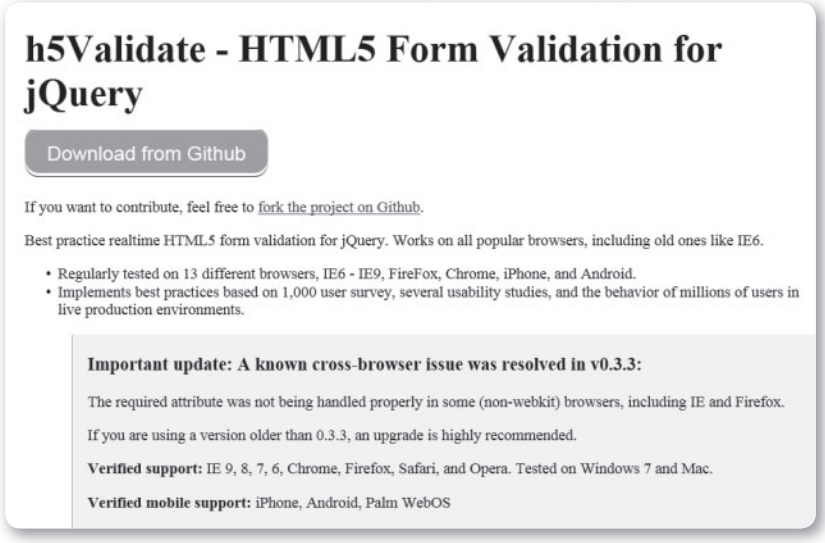


Figura 20. Si buscamos opciones de validación especialmente desarrolladas para HTML5, podemos encontrar los beneficios que nos ofrece **`h5Validate`** (**`http://ericleads.com/h5validate`**).

Recordemos que el ejemplo desarrollado en este capítulo puede ser válido para otros proyectos que incluyan formularios; solo será preciso adaptarlo a cada necesidad proyectual.

RESUMEN

En este capítulo, hemos conocido la evolución de los formularios en la Web. Vimos las características principales con las que contamos en HTML4/XHTML. Aprendimos cómo potenciarlos con JavaScript y las técnicas que introdujo AJAX. Luego, analizamos las características que introduce HTML5 para la creación de formularios. Para finalizar, vimos cómo aplicar lo aprendido en un ejemplo concreto de un formulario de contacto, partiendo desde HTML4 y agregando validación con AJAX y, luego, cómo incorporar las características de HTML5. Además, nos encargaremos de cerrar con las recomendaciones para crear un formulario con nuevas tecnologías desde cero.

Actividades

TEST DE AUTOEVALUACIÓN

1. Indique los atributos que puede recibir la etiqueta **<form>**.
2. ¿Qué diferencias hay entre el uso de **get** y **post** en la acción del formulario?
3. ¿Cómo se define un campo de entrada de un formulario para contraseñas?
4. ¿Para qué se utiliza la etiqueta **<optgroup>**?
5. ¿Qué valores se le pueden asignar al atributo **type** de una etiqueta **<button>**?
6. ¿Para qué se utiliza el atributo **placeholder** y en qué elementos se puede aplicar?
7. ¿Qué opciones existen para definir campos numéricos en HTML5?
8. ¿Cuáles son los cambios que se introducen en HTML5 para **<button>**?
9. ¿Qué atributos se agregan para **<textarea>** en HTML5?
10. ¿Cuáles son los cambios que se le aplican a la etiqueta **<legend>** en HTML5?

EJERCICIOS PRÁCTICOS

1. Cree un formulario de contacto en HTML4/XHTML.
2. Utilice una librería de JavaScript/AJAX para validar el formulario creado en el punto anterior.
3. Agregue, al formulario creado, la opción de autofocus de HTML5 en el primer campo y reemplace la validación de campo requerido por la opción que brinda HTML5.
4. Cree la estructura de un formulario de búsqueda con HTML5.
5. Diseñe desde cero un formulario HTML5 para que el usuario pueda registrarse. Utilice campos de e-mail, dirección web y teléfono. Incluya autofocus en el primer campo y agregue la característica **placeholder** en todos los campos.

*07 –

Diseño y desarrollo web adaptado a móviles

En este capítulo, veremos los aspectos que debemos tener en cuenta acerca del panorama actual para diseñar y desarrollar soluciones web para móviles. Nos introduciremos en las técnicas de detección de características y pantallas de dispositivos móviles. Además, conoceremos las ventajas que introduce HTML5 en este rubro, y veremos las herramientas y los frameworks que nos ayudarán en el desarrollo y testeo.

- **La evolución de la Web móvil . 242**
- **Ventajas de HTML5 para las plataformas móviles 244**
- **Navegadores para móviles 245**
  - Mobile Safari..... 245
  - Navegador Android 246
  - Internet Explorer Mobile..... 247
  - Firefox para móviles 248
  - Opera Mini y Opera Mobile..... 248
- **Cómo saber si el móvil está preparado para HTML5 ..... 251**
- **Herramientas de desarrollo web para móviles 256**
- **Crear aplicaciones web..... 258**
- **Resumen 261**
- **Actividades 262**

La evolución de la Web móvil

La navegación web a través de móviles se ha convertido en una tendencia que se mantiene en ascenso constante. Cada vez son más y de mayor potencia los dispositivos que se comercializan, lo que ha generado un gran crecimiento del mercado.

Las tecnologías relacionadas con este fenómeno también han evolucionado en el último tiempo, y las técnicas de desarrollo acompañan estos cambios. Desde las primeras especificaciones del lenguaje **WML** (*Wireless Markup Language*) y del estándar **WAP** (*Wireless Application Protocol*), en la primera época de la Web móvil, hasta el soporte actual de XHTML, HTML5, CSS3 y JavaScript por parte de muchos navegadores **mobile**, un largo camino se ha transitado y una importante evolución se ha producido tanto en la Web como en los dispositivos.

En la actualidad, gran parte de los aparatos que cuentan con tecnologías para navegar por Internet pueden hacerlo sin inconvenientes por la mayoría de los sitios que se encuentran disponibles hoy en día.

Figura 1. Existen varios sitios que ofrecen desarrollos para que sean vistos por móviles, como **`www.google.es/mobile`**.

Respecto de la compatibilidad con móviles, debemos tener en cuenta que algunas de las tecnologías que se utilizan en el desarrollo web no son compatibles en todas las plataformas móviles. Un ejemplo de esto pueden ser Flash y Silverlight, que son soportados solo por algunos de los navegadores mobile disponibles en el mercado.

> LA DETECCIÓN DE LAS CARACTERÍSTICAS SOPORTADAS POR EL NAVEGADOR MÓVIL ES FUNDAMENTAL.

En lo que se refiere a la interfaz de usuario, existe un cambio de paradigma muy importante. Las tecnologías **touch** y **multi-touch** marcan un cambio también para el diseño de interfaces. El mouse deja su papel protagónico a los dedos, y esto nos obliga a repensar la manera de crear los elementos accesibles que componen la interfaz de la Web y también el uso de los eventos, ya que para este tipo de dispositivos no tiene sentido el uso de **onmouseover** u **onmouseout**, por ejemplo.

Figura 2. iPhone (**www.apple.com/es/iphone**) es el smartphone desarrollado por Apple que se destaca por su pantalla de alta calidad y soporte multi-touch, y también por la compatibilidad con HTML5 en su navegador.

También es importante tener en cuenta que los usuarios que acceden desde un móvil o una tablet a un sitio o a una aplicación web esperan contar con una versión optimizada para su plataforma y, de ser posible, acceder a una experiencia pensada para aprovechar al máximo los recursos que pueden ofrecer estos tipos de dispositivos.

Ventajas de HTML5 para las plataformas móviles

En el ámbito del desarrollo mobile, uno de los debates que se plantean es el desarrollo de **aplicaciones nativas** o **aplicaciones web**.

Las aplicaciones nativas tienen como ventaja la personalización que puede realizarse para el medio y el acceso al hardware. Sin embargo, por parte de las soluciones web, estos aspectos pueden superarse en la actualidad con la evolución de las técnicas de desarrollo y las nuevas posibilidades que ofrecen a móviles, incluida la opción de que el navegador tenga acceso al hardware en ciertos modelos.

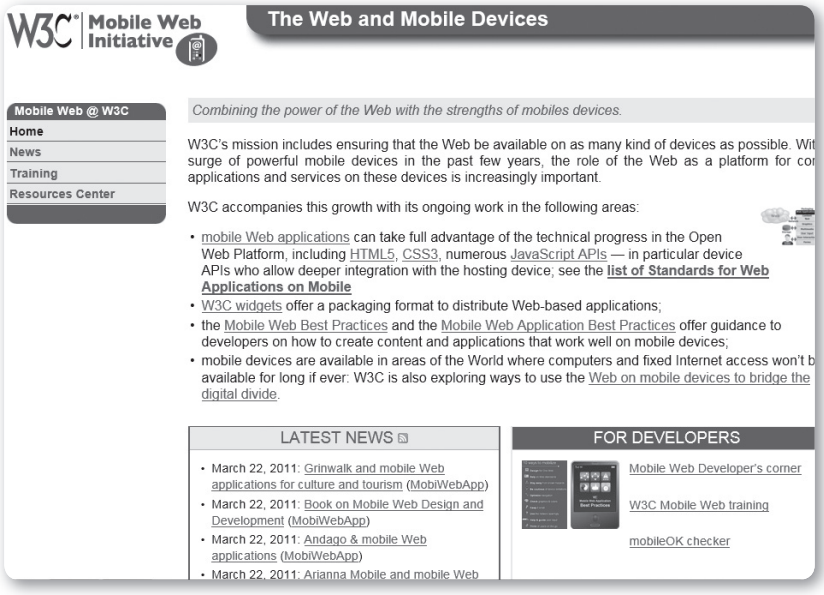


Figura 3. W3C tiene en cuenta los dispositivos móviles, como vemos en la dirección que se encuentra en **`www.w3.org/Mobile`**.

Algunas de las características de HTML5 que pueden resultar especialmente útiles para el desarrollo de móviles tienen que ver con las posibilidades multimedia para audio y video, acceso a la API de geolocalización y características de formulario. Otro de los aspectos

que abren un interesante camino para el desarrollo en móviles es la posibilidad de crear aplicaciones web que puedan funcionar **offline**. También algunas de las características que incorpora CSS3 comienzan a ser soportadas por ciertos navegadores móviles.

Navegadores para móviles

Existen diversos sistemas operativos para móviles. Algunas compañías, como el caso de Apple, han desarrollado sus propios sistemas para dispositivos de este tipo. Otras empresas han optado por soluciones que puedan adaptarse a diferentes clases de dispositivos, como el claro ejemplo de Android. Si bien la mayoría de los sistemas móviles que se encuentran preparados para conectarse a Internet cuentan con un navegador y software adecuado para hacerlo, también existen compañías que desarrollan versiones de browsers multiplataforma, como por ejemplo, Opera.

A continuación, repasaremos los principales navegadores para plataformas móviles y sus características.

Mobile Safari

Este navegador es el que utilizan los dispositivos móviles desarrollados por Apple. Basado en WebKit, cuenta con muy buenas opciones de zoom, interfaz multi-touch para acceder mediante gestos de los dedos, y muy buena compatibilidad con características de HTML5 y CSS3. Su motor JavaScript es Nitro, y se destaca su desempeño en el Acid3 test.

Distribuido junto al sistema iOS, ofrece soporte para características como geolocalización, almacenamiento del lado cliente, FormData (XMLHttpRequest Level 2), video (HTML5), formularios avanzados

WML (WIRELESS MARKUP LANGUAGE)

Wireless Markup Language, más conocido como **WML**, es el nombre de un metalenguaje pensado para móviles que se conectan mediante WAP y que se basa en XML. En el último tiempo, este lenguaje ha perdido protagonismo, ya que los móviles han avanzado mucho sobre el soporte a XHTML, y también han comenzado a aceptar HTML5 y otros estándares web.

(HTML5), WebSockets (HTML5), Canvas (HTML5) y SVG (HTML5). Vale recordar que este navegador no soporta Flash.

Una característica interesante que se le ha incorporado a este producto (desde iOS 4.2) es la posibilidad de acceder al acelerómetro y giroscopio del móvil desde el navegador, mediante código programado en JavaScript, gracias al acceso a la Device Orientation API y a los eventos de movimiento (*Device Motion Event*).

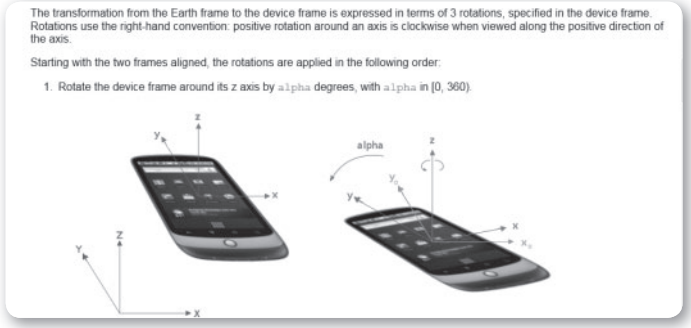


Figura 4. Para revisar Device Orientation y Device Motion vamos a **http://dev.w3.org/geo/api/spec-source-orientation.html**.

Podemos encontrar más información para el desarrollo de soluciones web enfocadas en este navegador en **Safari Developer Library** (**http://developer.apple.com/library/safari/navigation**).

Navegador Android

Debemos tener en cuenta que el navegador que se incluye junto a los sistemas operativos Android (conocido como **Android Web Browser**) dispone de un motor basado en WebKit, y también debemos saber que V8 es su motor de JavaScript. Cabe destacar que este software, al igual que Chrome, son desarrollos de Google.

Este navegador móvil soporta HTML5 y logra buen desempeño en los tests, pero vale aclarar que no soporta todas las característias de Chrome.

Este navegador posee soporte para acceso a la API de geolocalización, acceso a hardware (acelerómetro), animaciones con Canvas de HTML5, Web Storge, elementos de formulario HTML5 y estilos con CSS3, entre

otras muchas funcionalidades accesibles desde este browser. Es importante aclarar que este navegador puede soportar el reproductor de Flash (dependiendo de la versión del sistema y el hardware).

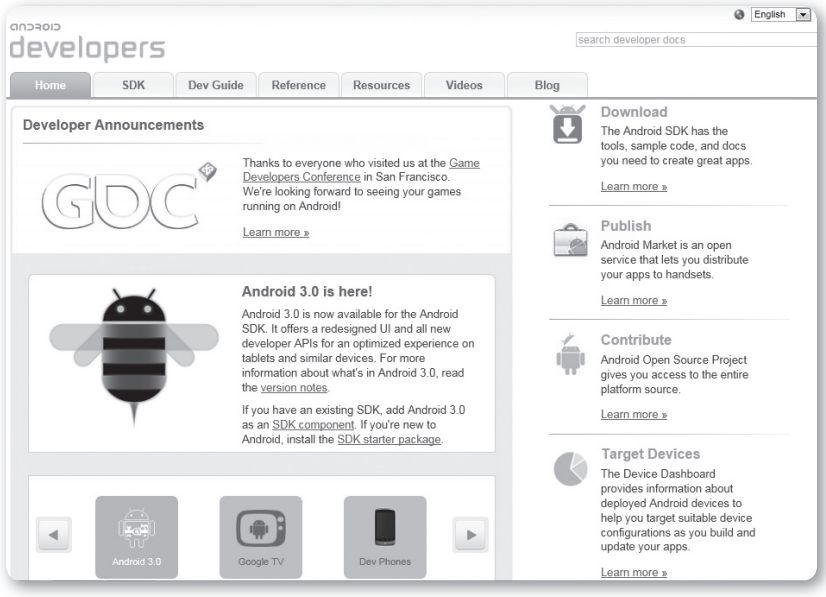


Figura 5. Podemos saber más sobre Android para desarrolladores ingresando en **`http://developer.android.com/index.html`**.

Internet Explorer Mobile

Este navegador, en sus primeros tiempos en la década del noventa, fue conocido como Pocket Internet Explorer y se distribuyó con los sistemas para móviles Windows CE. Originalmente, no utilizaba el mismo motor que la versión de escritorio.

En la actualidad, funciona como navegador predeterminado en Windows Phone 7, utiliza motor Trident, soporta interfaces multitáctiles y tiene diferentes niveles de zoom.

A partir de Internet Explorer Mobile 9, se incorpora soporte a HTML5 y aceleración por hardware, funciones que se pueden apreciar en la versión de escritorio de IE9. Este navegador se incluye junto a Windows Phone 7, que podemos encontrar en el sitio web que está en la dirección **www.microsoft.com/windowsphone/es-es/default.aspx**.

Firefox para móviles

Mozilla ha introducido una versión de Firefox pensada especialmente para móviles, cuyo nombre es **Firefox for mobile**. Este producto, que es una evolución del que anteriormente se denominaba **Fennec**, hoy por hoy funciona en sistemas Maemo, Windows Mobile 6.0 Professional (o superior) y Android (en este último caso, dependiendo del dispositivo).

Figura 6. Podemos obtener Firefox para móviles ingresando en `www.mozilla.com/es-ES/mobile`.

Este navegador para móviles cuenta con el mismo motor de render que Firefox (Gecko), y su motor de JavaScript es JaegerMonkey. Ofrece compatibilidad con HTML5 (Web workers, Offline storage, Canvas y SVG, entre otras características), CSS3 y JavaScript.

Opera Mini y Opera Mobile

Es importante recordar que Opera es una compañía que, entre sus productos, se destaca en especial por ofrecernos, de manera completamente gratuita, versiones de su navegador para diferentes plataformas, entre las que se encuentran las siguientes:

Opera Mini es una alternativa muy interesante que se basa en un motor de render online (ubicado en los servidores de Opera) para ofrecer un flujo de datos optimizado. Es posible probar una demostración de un simulador de Opera Mini, el cual encontramos en el sitio web que está en la dirección **www.opera.com/mobile/demo**. Según las características de nuestro dispositivo, es posible optar por **Opera Mobile**, un completo

y potente navegador para móviles, que utiliza el mismo motor que la versión de escritorio (Presto), y que, incluso, brinda soporte para HTML5 y acceso al acelerómetro (solo obtendremos acceso a esta característica si está disponible en el teléfono correspondiente).

Figura 7. Para obtener alguna de estas versiones, podemos ingresar en **`www.opera.com/mobile`**.

Otros navegadores para móviles

Además de los navegadores mencionados anteriormente, podemos encontrar otros, entre los que se destacan:

- **BlackBerry Browser**: navegador de los dispositivos BlackBerry. Utilizó el motor de render Mango entre la versión 4.5 y la 5, pasando a WebKit desde la versión 6. En el sitio oficial, podemos encontrar el SDK, y simuladores que permiten realizar desarrollos y pruebas destinadas a esta plataforma móvil.

HISTORIA DE SYMBIAN

El sistema operativo **Symbian** fue creado para funcionar en diversos teléfonos móviles. En un principio se distribuía junto a una licencia propietaria, pero posteriormente pasó a ser código abierto (luego de algún tiempo comenzó a entregarse junto a una licencia EPL). En la actualidad, es mantenido por la empresa Nokia y, a lo largo de su historia, diferentes compañías lo han utilizado para algunos de sus teléfonos, entre los cuales podemos mencionar a las siguientes: Sony Ericsson, Motorola y BenQ, además de otras menos conocidas Encontramos más información en: **http://symbian.nokia.com**.

- **NetFront**: desarrollado por la empresa japonesa Access Co. Ltd., funciona en una gran variedad de plataformas. Su motor es NetFront. Soporta HTML 4.01, algunas características de HTML5 (según la versión) y también AJAX. Más información en **www.access-company.com/products/mobile_solutions/netfrontmobile/browser/index.html**.

Figura 8. BlackBerry ofrece simuladores en: **`http://us.blackberry.com/developers/browserdev`**.

- **Blazer**: creado por Palm para sus sistemas, está basado en NetFront. Soporta HTML 4.01/XHTML 1.0, WML 1.3 y JavaScript, entre otras características. Podemos ver más sobre sus particularidades en el sitio **www.palm.com/ar/mobilemanagers/lifedrive/blazer.html**.

Vale destacar que **Symbian**, uno de los sistemas móviles más usados, se encuentra entre los precursores del uso de WebKit en su navegador para móviles. El navegador de Symbian Anna es compatible con HTML5 y CSS3.

EVOLUCIÓN DE IOS DE APPLE

Bajo el nombre de **iOS**, se conoce el sistema operativo desarrollado por Apple para sus dispositivos móviles. Este sistema es utilizado tanto por iPhone como también por iPod Touch e iPad. Originalmente, fue llamado iPhone OS, ya que se comenzó a utilizar en el smartphone de Apple, para luego extenderse a otros productos de la compañía.

Cómo saber si el móvil está preparado para HTML5

Debemos saber que, al igual que en un navegador de escritorio, en el caso de los móviles podremos encontrar un variado nivel de compatibilidad con HTML5 y CSS3.

De la misma manera que ocurre con las opciones desktop, es posible trabajar con JavaScript para detectar compatibilidad.

Modernizr (que podemos encontrar en la dirección web **www.modernizr.com**) es una librería sobre la que ya hemos hablado. También puede resultarnos útil en la detección de las características con las que cuenta el navegador del móvil.

Figura 9. Si ingresamos en `www.modernizr.com/releases`, podremos conocer las últimas actualizaciones de Modernizr.

Técnicas de detección

Existen diversas formas para detectar si el usuario que está accediendo a nuestro sitio web lo hace desde una plataforma móvil o si, por el contrario, se maneja desde una de escritorio. En cualquiera de los casos, también dependeremos de ciertas características del navegador del móvil para poder tener éxito con este proceso.

Si bien existen diferentes soluciones mediante JavaScript o,

incluso, empleando lenguajes del lado servidor, en esta oportunidad veremos las opciones que nos ofrece CSS, por medio de la regla **@media** y Media Queries, característica incorporada a partir de CSS3.

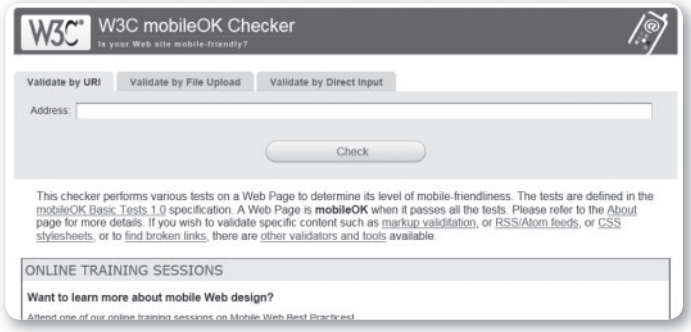


Figura 10. Para saber si nuestro sitio web es mobile-friendly, podemos usar **`http://validator.w3.org/mobile`**.

Detección del dispositivo con CSS

Al trabajar con hojas de estilo, podemos detectar el tipo de dispositivo y actuar en consecuencia para establecer determinadas reglas según las necesidades de diseño o desarrollo de nuestro proyecto.

UNA WEB COMPATIBLE CON MÓVILES NOS LLEVA A EVALUAR NUEVOS FACTORES EN EL DESARROLLO.

La regla **@media** es la que permite definir diferentes reglas (valga la redundancia) para los selectores que deseemos.

En el **Capítulo 4**, veíamos que **@media** en CSS 2.1 puede soportar diferente tipos, entre ellos **screen** (para pantalla) y **handheld** (para dispositivos de mano).

Podremos establecer diferentes medidas para un título, dependiendo del medio donde sea representado. Veamos un ejemplo para nuestras hojas de estilos en **screen** y **handheld**:

```
@media screen {h1 {font-size:24px;}}
@media handheld {h1 {font-size:14px;}}
```

Siempre debemos tener en cuenta que esta regla actúa sobre la base del **User Agent** con el que se presente el navegador del usuario (o medio). Recordemos, también, que el navegador, al acceder a una página, entrega al servidor (mediante una cadena de texto) la información de User-agent, que entre otros datos puede contener nombre del software con el que se está accediendo, versión, sistema operativo e idioma. Esto se puede detectar mediante programación del lado servidor o scripts.

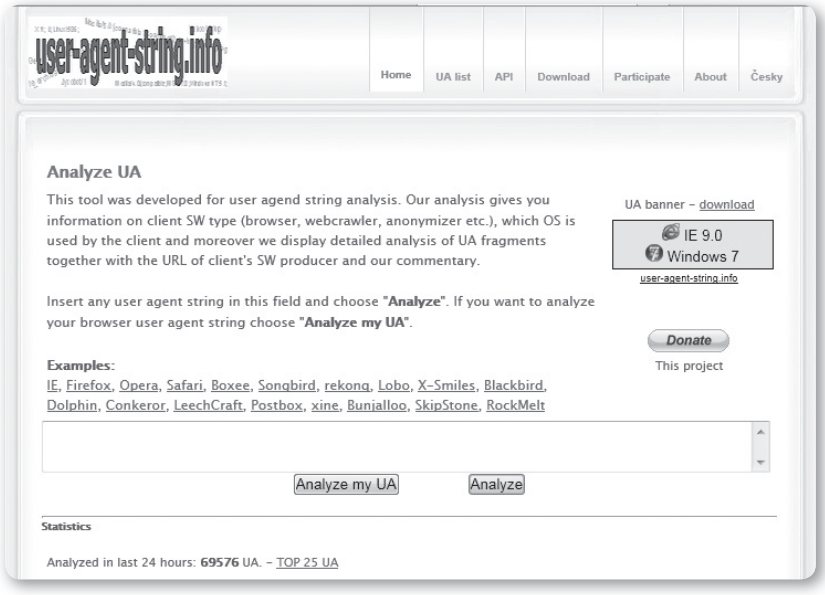


Figura 11. Para verifica el User Agent con el que se presenta nuestro navegador, accedemos a **`http://user-agent-string.info`**.

En el último tiempo, con la evolución de las tecnologías móviles, cada vez podemos encontrar mayor cantidad de variantes de agentes, y, en algunos casos, puede ocurrir que no sea posible detectar con eficiencia el tipo de medio con la regla **@media**. Buenos ejemplos son iPhone y otros productos móviles de Apple, que utilizan Safari Mobile con agente definido como **screen**. Por esta razón, en CSS3 se ha incorporado **Media Queries**, una característica que nos ayudará con la detección de una manera más eficaz.

CSS3 Media Queries

Con el éxito y la masificación de los dispositivos móviles, se introduce **Media Queries** en CSS3 para permitir realizar una detección más precisa de la pantalla del medio con el que el usuario está accediendo al sitio web, y poder actuar sobre la base de las necesidades del desarrollo y del diseño.

A diferencia de la opción de detección que veíamos anteriormente, con Media Queries lo que podremos hacer es mostrar diferentes reglas de estilos según características de pantalla relacionadas con su alto y su ancho. De esta manera, podríamos tener una estructura definida en un documento HTML y, mediante estas opciones, ofrecer diferentes hojas de estilos, que dependerán de la pantalla del usuario que accede.

A continuación, veamos las opciones que tenemos para trabajar con esta característica:

- **width**: ancho del área de destino en la ventana del dispositivo, soporta los prefijos **max** y **min** (máximo y mínimo), por ejemplo: **max-width** o **min-width**.
- **device-width**: ancho de la superficie de render del dispositivo. Soporta los prefijos **max** y **min**.
- **height**: alto del área de destino en la ventana del dispositivo. Soporta **max** y **min**.
- **device-height**: alto de la superficie de render del dispositivo. Soporta los prefijos **max** y **min**.
- **orientation**: orientación del dispositivo; puede recibir el valor **portrait** o **landscape**. No soporta **max** y **min**.
- **aspect-ratio**: es el ratio definido en el medio por **width** y **height**. Soporta **min** y **max**.
- **device-aspect-ratio**: es el ratio definido en el medio por **device-width** y **device-height**. Soporta **min** y **max**.

PANTALLAS DE SMARTPHONES Y TABLETS

Los teléfonos inteligentes y las tablets tienen diferencias en sus características técnicas conocidas, pero también existen coincidencias. En lo referente al desarrollo web, si utilizamos técnicas de detección basadas en tamaño de pantalla, debemos tener en cuenta y considerar el mayor tamaño de las tablets, y ofrecer un desarrollo y recursos optimizados para cada uno de los medios.

- **color**: es el número de bits de color del dispositivo. Si no es un dispositivo color, el valor es **0**. Soporta **max** y **min**.
- **scan**: se emplea para medios del tipo **tv** y puede tener el valor **progressive** o **interlace**.
- **resolution**: es la resolución del medio, puede recibir un valor de longitud y trabajar con los prefijos **min** y **max**.
- **monochrome**: se utiliza para medios visuales, recibe un valor de longitud y se utiliza para definir la cantidad de bits por píxel en monocromo. Si el dispositivo no es monocromo, el resultado de salida resultará **0**.

Como podemos notar, de estas características nos pueden resultar de mayor utilidad aquellas que nos permiten acceder al ancho y alto, y también a la orientación. A continuación, veremos un ejemplo:

```
<link rel="stylesheet" media="all and (max-device-width: 480px)" href="móvil_principal.css" type="text/css" >
<link rel="stylesheet" media="all and (min-device-width: 481px)" href="otros_principal.css" type="text/css" >
```

Con el código que hemos visto, especificamos un archivo CSS para dispositivos que tengan pantallas de hasta 480 px y otro distinto para aquellos que tengan pantallas desde 481 px.

Si deseamos emplear Media Queries en nuestros desarrollos web, debemos analizar algunos aspectos. En primer lugar, contemplar que resultará preciso definir diseños diferenciados, estableciendo hojas de estilos distintas. Será necesario analizar si los dispositivos a los que apuntamos lo hacen. Por ejemplo, Safari Mobile y el navegador de Android soportan esta característica.

MEDIA QUERIES PERMITE UNA EFICIENTE DETECCIÓN EN NAVEGADORES MÓVILES MODERNOS.

También resulta importante contemplar el tamaño de pantalla máximo para los móviles, algo que puede estar más claro en los equipos de Apple, pero quizás en la diversidad de modelos que soportan Android esto no resulte tan fácil de determinar. Aquí, debemos también tener en cuenta si haremos diferenciación entre desktop, tablets y smartphones, contemplando la diversidad de pantallas que pueden encontrarse entre estos dispositivos.

Son varios los factores que debemos contemplar tanto para el diseño, como para la optimización de recursos (por ejemplo, de imágenes) que utilizaremos para mostrar en cada dispositivo.

Finalmente, tengamos en cuenta que adaptar un diseño y un desarrollo web a una plataforma móvil no es solo "hacerla más pequeña". Muchos otros aspectos entran en juego, tales como accesibilidad, navegabilidad, usabilidad y aprovechar al máximo las nuevas características que ofrecen los dispositivos móviles. Podremos conocer la especificación completa en **www.w3.org/TR/css3-mediaqueries**.

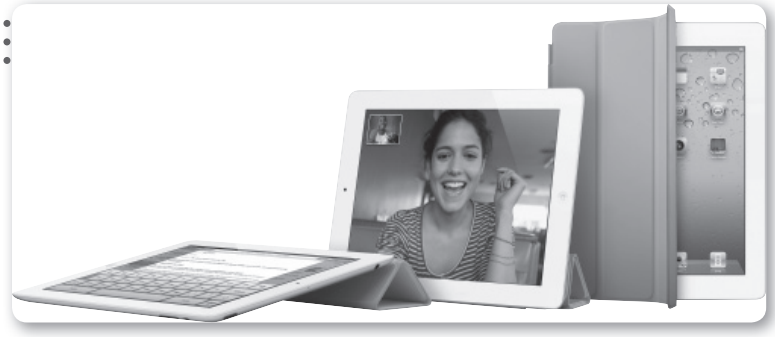


Figura 12. Tanto tablets como móviles pueden cambiar su orientación, modificando su ancho máximo disponible (imagen cortesía de Apple).

Herramientas de desarrollo web para móviles

Como hemos podido ver hasta aquí, el desarrollo web para móviles puede utilizar lenguajes de etiquetas, hojas de estilos y características de AJAX que, perfectamente, es posible escribir en cualquier editor de texto enfocado a código o incluso en programas WYSIWYG, como Dreamweaver (en sus nuevas versiones incluye funciones especificas para móviles).

Una opción muy útil para el desarrollo de soluciones web para móviles es contar con un dispositivo físico para poder probar todas las funcionalidades como lo haría el usuario final. Otras herramientas que nos

pueden resultar de mucha ayuda son los diferentes emuladores y simuladores de móviles que, en numerosas ocasiones, se ponen a disposición de los desarrolladores a través de los **SDK** (*Software Development Kit*) o diferentes paquetes de herramientas. Si buscamos soluciones para **Android**, tenemos la posibilidad de descargar de manera gratuita el **Android SDK**, que se encuentra disponible para Linux, Mac y PC en el sitio que encontramos en la dirección **http://developer.android.com/sdk/index.html**.

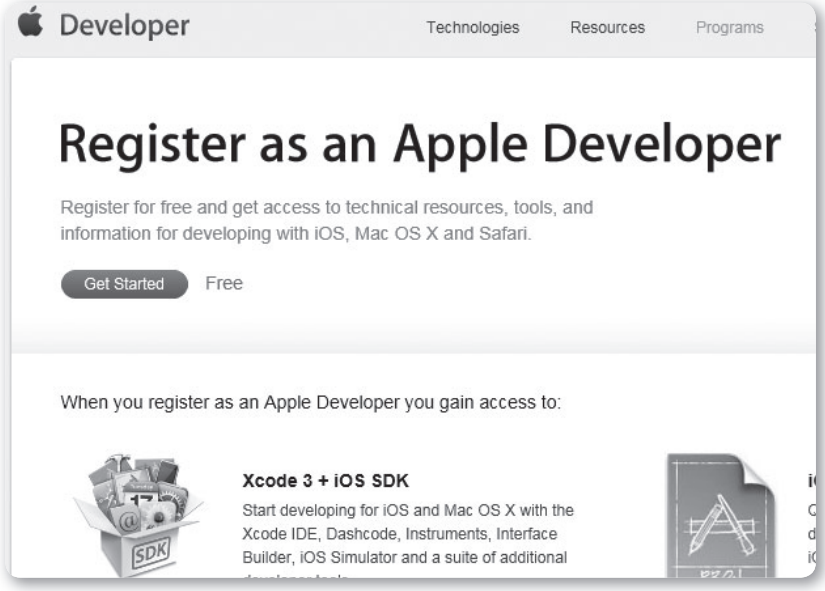


Figura 13. En **`http://developer.apple.com/programs/register`**, podremos descargar el **SDK** de **iOS**.

Si buscamos herramientas para desarrollo de **webOS**, podemos ingresar en **http://developer.palm.com**. Allí encontraremos guías, acceso a la comunidad, noticias de eventos y también la posibilidad de ingresar en la sección **Tools**, donde podremos obtener el HP webOS Platform SDK/PDK.

Por parte del navegador de **Mozilla** para móviles, si ingresamos en **www.mozilla.com/en-US/mobile/download**, además de descargar el programa para nuestro teléfono podremos obtener una versión para

desarrolladores para equipos de escritorio con sistemas Windows, Mac OS X o Linux.

Es necesario tener en cuenta que Opera Mobile Emulator puede ser descargado visitando el sitio web que encontramos en la dirección **www.opera.com/developer/tools**, donde haremos clic sobre el enlace adecuado.

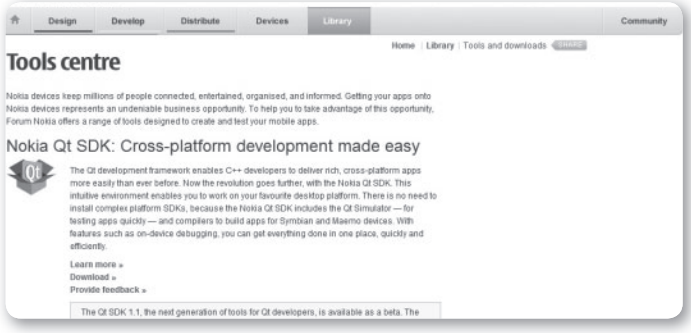


Figura 14. Encontramos el **SDK** de Nokia en **`www.forum.nokia.com/Library/Tools_and_downloads`**.

Crear aplicaciones web

Con el éxito de la Web móvil y el auge de HTML5, han surgido librerías que aprovechan estas características, sumadas a otras técnicas, para ofrecer soluciones eficaces que nos ayuden a resolver nuestros desarrollos con mayor flexibiliad y sin tantas complicaciones.

La creación de aplicaciones web para móviles empleando HTML5

HISTORIA DE ANDROID

Con sede en California, Estados Unidos, **Android Inc.** desarrolló un sistema operativo para móviles basado en Linux al que denominó **Android**. Este proyecto fue adquirido por Google en el año 2005. Tres años después, se lanzaría la primera versión pública de este sistema. En la actualidad, Android es utilizado por una gran variedad de móviles y tablets.

puede ser potenciada gracias al uso de diferentes frameworks, como veremos a continuación.

Frameworks para desarrollo de soluciones para móviles

Sencha Touch es un framework basado en JavaScript y HTML5, pensado para el desarrollo de aplicaciones web para móviles, que aprovecha las características de los dispositivos, entre los que se incluyen Android, iPhone y BlackBerry.

Este producto cuenta con licencia Open Source o Comercial, según sea el uso que se haga de él. Podemos conocer más sobre las opciones de licencia en **www.sencha.com/products/touch/license**.

Figura 15. En **www.sencha.com/products/touch**, conoceremos las opciones de este framework y podremos descargarlo.

The-M-ProJect es un framework que se basa en JavaScript y HTML5 para brindar soluciones cross-platform en el desarrollo de aplicaciones móviles. Este framework, que puede usarse con plataformas iOS, Android, Palm webOS y BlackBerry, cuenta con licencia MIT (Open Source). En el sitio web de este framework podremos encontrar una completa documentación, tutoriales y también diferentes ejemplos que nos ayudarán a comprender su funcionamiento y las posibilidades que nos ofrece para nuestros futuros desarrollos. Si deseamos conocer las novedades de este producto podemos seguir la cuenta de Twitter @_themproject.

Figura 16. En el sitio web **`www.the-m-project.org`**, podemos leer más sobre este framework y ver ejemplos de uso.

DHTMLX Touch es un framework para el desarrollo de aplicaciones web destinadas a dispositivos móviles tactiles.

Este framework se encuentra disponible en diferentes productos y se puede obtener de manera gratuita o paga, según las necesidades del desarrollo. Podemos obtener más información sobre los tipos de licencia en **www.dhtmlx.com/docs/products/licenses.shtml**.

Figura 17. Si ingresamos en **`www.dhtmlx.com/touch`**, podremos acceder a la documentación y descarga de esta librería.

jQTouch es un plugin de jQuery pensado para desarrolladores que necesitan crear aplicaciones web para iPhone o iPod Touch, entre otros dispositivos compatibles con estas tecnologías. El propósito de esta

librería es brindarnos una solución con una gran potencia, incluso, para poder simular interfaces que se asemejen a las aplicaciones nativas de los mencionados dispositivos. Este plugin nos facilita el trabajo con las características de WebKit móvil; incluye opciones de animación y, por supuesto, soporte a capacidades de HTML5 (especialmente en lo referente a formularios).

Figura 18. jQTouch se puede descargar desde el sitio que encontramos en la dirección **`http://jqtouch.com`**.

Otra alternativa es la que brinda PhoneGap (www.phonegap.com), un framework que permite desarrollar aplicaciones para móviles compatibles con las plataformas más populares, aprovechando las características de HTML5, CSS3 y JavaScript.

RESUMEN

En este capítulo, hemos hablado sobre las tecnologías móviles, cada vez más utilizadas por los usuarios en todo el mundo. Vimos su evolución y cómo entra en juego HTML5. Conocimos los principales navegadores móviles y sus características más importantes. Analizamos técnicas para detectar características del dispositivo con frameworks, vimos lo que nos ofrece CSS 2.1 con @media y, luego, conocimos Media Queries de CSS3. Finalmente, analizamos qué nos brindan algunos framewoks de AJAX para trabajar con JavaScript y HTML5 en el desarrollo de soluciones web para móviles.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué diferencias más destacadas encuentra entre el desarrollo web para móviles y el desarrollo web para desktop?
2. ¿Qué motor de renderizado utiliza Safari Mobile?
3. ¿Cuál es el motor de JavaScript de Safari Mobile y cuál es el que utiliza el navegador de Android?
4. Mencione algunas de las características de HTML5 que se pueden utilizar con iPhone.
5. Explique las diferencias entre Opera Mini y Opera Mobile.
6. Indique cómo se puede detectar si el navegador del móvil puede aceptar características de HTML5 y CSS3.
7. ¿Cuáles son las ventajas que introduce Media Queries de CSS3?
8. Mencione, al menos, dos herramientas que faciliten el desarrollo web para móviles.
9. Describa las ventajas que ofrece el framewrok Sencha Touch.
10. ¿Qué es jQTouch y para qué se utiliza?

ACTIVIDADES PRÁCTICAS

1. Incorpore Modernizr a su proyecto para detectar si el navegador del móvil que accede soporta características de audio y video de HTML5.
2. Realice una visualización de un sitio en uno de los simuladores de móviles visto en este capítulo.
3. Utilice @media para detectar el medio y aplicar una hoja de estilos diferente para dispositivos que se presenten como handheld.
4. Emplee Media Queries para establecer hojas de estilos diferentes para dispositivos que tengan pantallas de hasta 320 px.
5. Descargue uno de los frameworks basados en HTML5 y JavaScript vistos en este capítulo, y cree un desarrollo que muestre contenido basado en texto e imágenes.

*08 –

Características avanzadas

Aquí analizaremos las características avanzadas que introducen HTML5 y las APIs relacionadas. Entre los temas que veremos se destaca el acceso al DOM, Canvas, SVG, WebGL, WebSockets, geolocalización, Drag & Drop nativo, Web Storage, Server-Sent Events, Web Workers, Application Cache API y el acceso a History API, entre otras.

- **JavaScript y DOM** 264
  - Seleccionar elementos del DOM 265
- **Canvas** 266
- **SVG** 274
- **WebGL (3D)** 277
- **WebSockets** 279
- **Geolocalización** 281
- **Drag & Drop** 288
- **Web Storage** 290
- **Web Messaging** 293
- **Server-Sent Events** 295
- **Web Workers** 296
- **Application Cache API** 298
  - Cache manifest 299
- **Resumen** 301
- **Actividades** 302

JavaScript y DOM

En este capítulo, al hablar de funciones avanzadas de HTML5, veremos varias características que se vinculan con las APIs de JavaScript. Los navegadores han comenzado a dar soporte nativo a algunas características relacionadas con diferentes APIs para brindar una mejor interacción con las aplicaciones desarrolladas.

A continuación, veremos algunas características que nos serán de gran utilidad para nuestros proyectos.

Figura 1. Ingresando en **`http://ie.microsoft.com/testdrive/HTML5/DOMStyle/Default.html`**, podremos interactuar con una demo que nos permite cambiar o remover los estilos del DOM.

API de selectores

Las **APIs de selectores** nos permiten ingresar un elemento y obtener como resultado el elemento seleccionado, pero como parte del DOM.

Si utilizamos **querySelector()**, obtendremos el primero de los elementos comparados; en cambio, si empleamos **querySelectorAll()**, contaremos con todo el conjunto de elementos que se encuentren. Ambas opciones pueden recibir selectores de CSS como parámetro.

Con **classList**, es posible verificar, agregar, remover o modificar una clase aplicada a un nodo del DOM. Entre los métodos que podemos usar, encontramos **contains**, **add**, **remove** y **toggle**.

La ventaja de esta característica es que ahora es una API nativa que se puede encontrar disponible en diversos navegadores.

Figura 2. Dojo (http://dojotoolkit.org) es una alternativa interesante para trabajar con características de AJAX y de HTML5.

Seleccionar elementos del DOM

Con AJAX, hemos aprendido varias técnicas muy importantes para el desarrollo web moderno. La posibilidad de recorrer el DOM y también de acceder a sus nodos de diversas maneras nos ha brindado mucha flexibilidad para incorporar nuevas funcionalidades en la creación de sitios o también de aplicaciones web.

Con HTML5, se introduce una API para trabajar con **getElementsByClassName()**; con esta opción, es posible recorrer el DOM para obtener los elementos que coincidan con la clase que hayamos

XMLHTTPREQUEST LEVEL 2

Actualmente en etapa de Working Draft, la primera versión de la especificación del nivel 2 de **XMLHttpRequest** se publicó en el año 2008. Entre la ventajas que introduce este nivel se destaca la posibilidad de trabajo cross-site y manejo de streams. Para conocer el estado del documento podemos ingresar en la dirección **www.w3.org/TR/XMLHttpRequest2**.

especificado. Con **getElementsByName()**, podremos pasar una cadena que contengan los valores del atributo **name** que deseemos encontrar.

La búsqueda dentro del DOM seleccionará aquellos elementos que contenga los valores indicados para el atributo **name**.

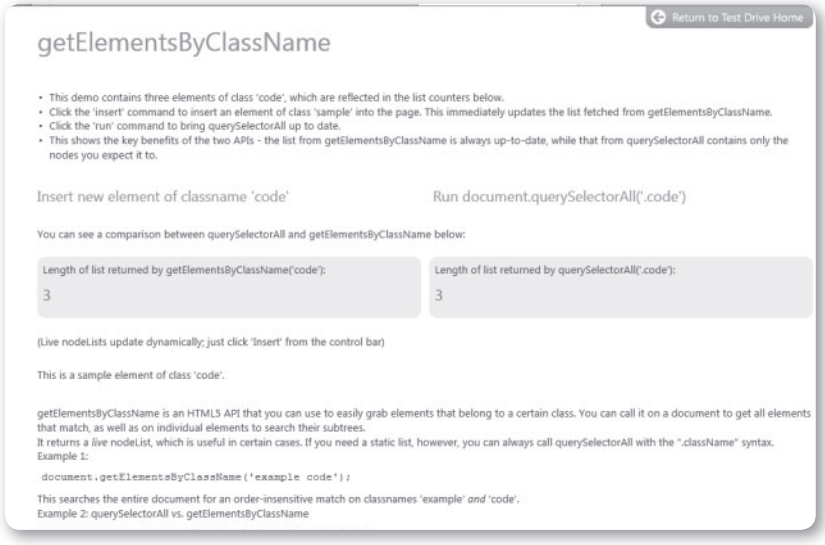


Figura 3. Podemos probar una demo que utiliza la API de HTML5 de getElementsByClassName si ingresamos en la dirección **http://ie.microsoft.com/testdrive.**

Para más información sobre este tema, recomendamos leer el documento que se ofrece en **www.w3.org/TR/html5/dom.html**.

Canvas

La incorporación de **Canvas** dota a HTML5 de la posibilidad de definir un área donde renderizar scripts que contengan diversas formas en dos dimensiones (rectángulos, arcos, líneas, curvas, etcétera) y también archivos de imágenes de mapa de bits. Además, se pueden utilizar efectos y transformaciones (rotación y escala, entre otras).

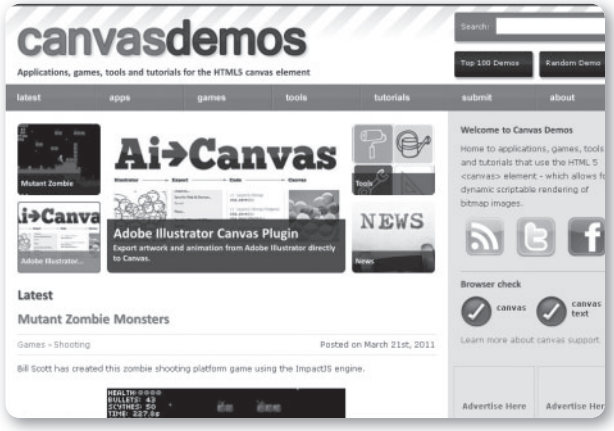


Figura 4. Al ingresar en el sitio **Web Canvas Demos** (**www.canvasdemos.com**), encontraremos ejemplos de uso de Canvas de HTML5.

El uso de la etiqueta **<canvas>** crea un nuevo universo de posibilidades. Esto se ve reflejado en desarrollos de aplicaciones, juegos y opciones multimedia, que no requieren soluciones creadas en otras plataformas, como el caso de Adobe Flash o Silverlight de Microsoft.

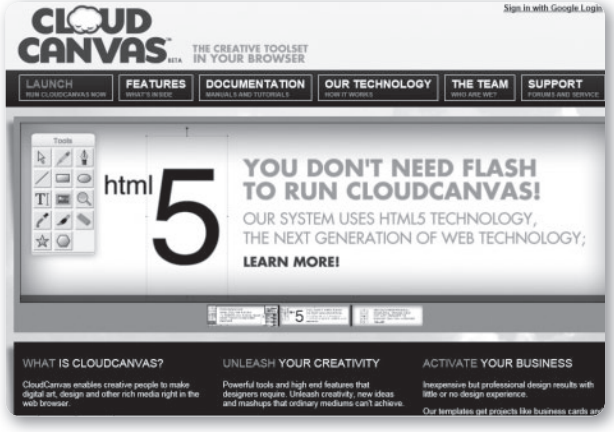


Figura 5. **Cloud Canvas** (en **www.cloud-canvas.com**) es una opción para realizar dibujos digitales. Se basa en AJAX y HTML5.

La etiqueta **<canvas>**, además de los atributos globales de HTML5, también soporta **height** (alto) y **width** (ancho), que pueden recibir un valor numérico expresado en píxeles.

Debemos tener en cuenta que la manera básica de definir un área para el elemento Canvas sería la que mostramos a continuación:

```
<canvas id="miCanvas" width="250" height="250">Su navegador no soporta
Canvas</canvas>
```

Este código mostrará el mensaje que escribimos para los navegadores no compatibles en caso de encontrarse con un browser que no soporte Canvas. En el resto de los casos, en sí mismo no mostrará nada en particular hasta que programemos el script que actuará con él. Veamos un ejemplo a continuación:

```
<script type="text/javascript">
        var canv=document.getElementById("miCanvas");
        var can2d=canv.getContext("2d");
        can2d.fillStyle="#FFFF00";
        can2d.fillRect(50,50,200,200);
        can2d.fillStyle="#CC0000";
        can2d.fillRect(10,10,100,100);
</script>
```

Este script, en primer lugar, realiza la declaración de variables y, luego, comienza con la acción. Primero se dibuja un cuadrado de color amarillo de 200 px de lado, desplazado a 50 x 50 px del origen. Por encima, se dibuja otro cuadrado, este de color rojo, que se encuentra desplazado 10 x 10 px del origen y mide 100 px de lado.

Para comprender el listado, veamos algunos de los atributos que podremos utilizar para realizar rellenos con Canvas:

- **fillStyle**: es el color del relleno y puede recibir como valor un color.
- **strokeStyle**: similar al anterior, pero se refiere al color del trazo.
- **lineWidth**: es una propiedad para indicar el grosor de una línea.

NO OLVIDAR LA COMPATIBILIDAD

Podemos darnos cuenta de que, al avanzar cada vez más en los contenidos propuestos en este libro, hemos profundizado sobre diversas mejoras que han sido introducidas por el lenguaje HTML5. Para nuestro trabajo será importante tener en cuenta que en todos los casos hay que contemplar soluciones de programación para que nuestros desarrollos ofrezcan compatibilidad con los principales navegadores de la actualidad, tanto para las plataformas desktop, como para las móviles de mayor penetración en el mercado actual.

- **lineCap**: permite establecer cómo se verán los extremos de una línea. Puede recibir los valores **butt** (predeterminado, es recto, arranca y termina exactamente con el path), **round** (redondeados, dibuja un semicírculo con base a partir del comienzo del path, superando esa posición) o **square** (dibuja un rectángulo a partir del inicio y final del path, superándolo).
- **lineJoin**: nos da la posibilidad de definir cómo se verán las líneas cuando se unen. Por ejemplo, en un cuadrado o en un rectángulo definirá cómo lucirán sus ángulos. Puede recibir los valores **miter** (predeterminado, no hay cambios), **round** o **bevel** (crea un biselado).

Resulta importante destacar que también es posible recurrir a algunos métodos de JavaScript para realizar el dibujo, hacer paths (trayectorias), mover, crear arcos y curvas. A continuación veremos el detalle:

- **getContext()**: es el método que invoca el contexto del Canvas. Debemos recordar que se utiliza en el comienzo.
- **fillRect()**: lo hemos visto en el ejemplo del listado que se ubica en la página anterior y sirve para dibujar rectángulos que tengan relleno. Puede recibir cuatro parámetros. Recordemos que los dos primeros corresponden a las coordenadas de los ejes X e Y.
- **strokeRect()**: permite dibujar el rectángulo sin relleno (solo el trazo del borde). Recibe los mismos parámetros que **fillRect()**.
- **clearRect()**: debemos tener en cuenta que con este método, podremos limpiar o borrar áreas de forma rectangular. Puede recibir los mismos parámetros que el método **fillRect()**.
- **beginPath()**: utilizamos este método para indicar que comenzará el dibujo de un path. No recibe parámetros.
- **moveTo()**: con este método, nos encargamos de mover el puntero que marca desde dónde comenzaremos a dibujar el path. Recibe como parámetros las coordenadas del eje X e Y.
- **lineTo()**: nos permite dibujar una línea. El origen será el lugar donde está posicionado el puntero. El final de la línea estará dado por los parámetros que le pasemos a este método (X e Y).
- **closePath()**: este método nos permite cerrar el path con una recta desde el punto inicial al final (no recibe parámetros).
- **fill()**: nos brinda la posibilidad de llenar con color el área que hayamos definido con el path. Si el dibujo realizado con el

path no está cerrado, se realizará el cierre con una línea recta (entre principio y final). Debemos recordar que no deben quedar segmentos intermedios sin dibujar porque, en ese caso, no tendrá efecto el relleno con este método.

- **stroke()**: se trata de una opción que nos permite realizar el dibujo de una línea en el contorno del path. A diferencia de **fill()**, este método no rellena, ya que solo dibuja el contorno.
- **arc()**: se emplea para dibujar arcos, con los cuales podremos describir una circunferencia (o un segmento de ella). Recibe como parámetros, primero, las coordenadas de los ejes X e Y. Los siguientes dos parámetros son los que corresponden al ángulo de comienzo y al de final, y deben estar expresados en radianes. El último parámetro que recibe es el sentido que tomará (**true** para sentido antihorario de principio a fin y **false** para el sentido contrario).
- **arcTo()**: nos permite redondear esquinas. Recibe los dos primeros parámetros como el X y el Y del subpath; los dos siguientes se refieren al X y al Y del final; el último valor es el radio.
- **bezierCurveTo()**: permite dibujar curvas Bezier, las cuales se definen a través de funciones matemáticas. El comienzo de la curva coincide con la posición inicial del puntero. Los dos primeros parámetros que le pasamos a este método nos permiten indicar los puntos X e Y del primer punto que define la tendencia de la primera curva. El siguiente par define el punto (X e Y) de la segunda curva. Los dos valores finales marcan el final, también con X e Y.
- **quadraticCurveTo()**: permite dibujar curvas cuadráticas, que se pueden considerar como un subtipo de curvas Bezier. La curva comienza a dibujarse desde donde esté parado el puntero; los

GURY

Dentro de las posibilidades disponibles para el desarrollo de soluciones que pueden prescindir de Flash, **Gury** es una interesante librería JavaScript que nos permite trabajar con la API Canvas, característica que se incorpora con la de la versión 5 de HTML. Este framework posibilita enfrentar la creación de desarrollos ahorrando una importante cantidad de código en su desarrollo. De esta forma, nos facilita en gran medida el camino para lograr aplicaciones funcionales utilizando un tiempo menor en el proceso de desarrollo. Podemos obtener esta librería y ver la documentación relacionada ingresando en **http://guryjs.org**.

primeros dos parámetros definen las coordenadas X e Y que tendría imaginariamente el punto que marca la inflexión en la tendencia tomada por la curva. Los dos parámetros siguientes indican los ejes X e Y donde concluye la curva.

Figura 6. Diversos proyectos han comenzado a ofrecer opciones basadas en Canvas, como **`http://mugtug.com/sketchpad`**.

A continuación, utilizaremos algunos de los métodos vistos hasta aquí para crear una forma triangular que se llenará de color azul:

```
<canvas id="miCanvas" width="150" height="150">Su navegador no soporta
Canvas</canvas>
<script type="text/javascript">
var canv=document.getElementById("miCanvas");
var canvastri=canv.getContext("2d");
if(canvastri){
  canvastri.fillStyle = 'blue';
  canvastri.beginPath();
  canvastri.moveTo(65,60);
  canvastri.lineTo(100,10);
  canvastri.lineTo(140,60);
  canvastri.closePath();
  canvastri.fill();
 }
</script>
```

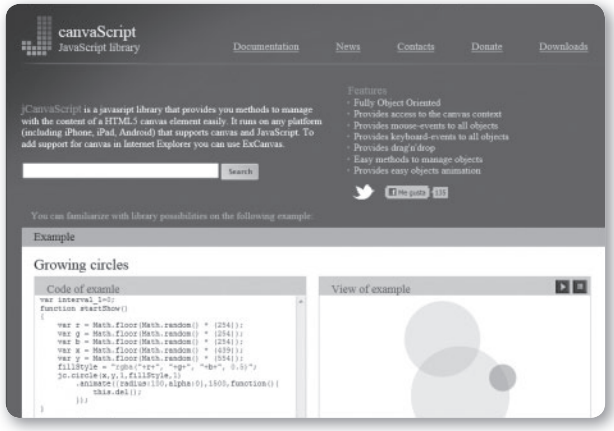


Figura 7. La librería **`jCanvaScript`** (en **`http://jcscript.com`**) nos permite trabajar con Canvas en forma sencilla.

Para trabajar con imágenes en Canvas de HTML5, contamos con el método **drawImage()**. Con esta opción, podremos mostrar imágenes (**GIF**, **JPG** o **PNG**) dentro de nuestro lienzo. Recibe tres parámetros; el primero es el objeto de la imagen que vamos a emplear, y los dos siguientes son su posición (en los ejes X e Y). Opcionalmente, podemos indicar dos parámetros más que indiquen el ancho y el alto de la imagen si deseamos escalarla. Si no indicamos estos dos últimos parámetros y la imagen entra en el espacio definido, no habrá problemas. En caso de que la imagen sea más grande que el espacio de nuestro Canvas, si no instrumentamos otras opciones y no la escalamos, no se mostrará.

Veamos un ejemplo de cómo aplicarlo:

```
<canvas id="miCanvas" width="640" height="480">Este navegador no soporta Canvas</canvas>
<script type="text/javascript">
        var canv=document.getElementById("miCanvas");
        var canvasimg=canv.getContext("2d");
        var imagen = new Image();
        imagen.src = 'ejemplo.jpg';
        imagen.onload = function(){
        canvasimg.drawImage(imagen,0,0,640,480);
}
</script>
```

En este listado definimos un área de Canvas de 640 x 480 px, cargamos una imagen instanciándola como un nuevo objeto en la variable imagen y, con **drawImage()**, la dibujamos a partir de las coordenadas X e Y en el ángulo superior izquierdo.

Además de los parámetros vistos, también se pueden agregar los de posición de lienzo (X e Y), y los de ancho y alto del lienzo.

Otra ventaja del elemento Canvas, además de las mencionadas, es que también soporta WebGL; esto abre la puerta a la posibilidad de dibujo en tres dimensiones empleando esta característica.

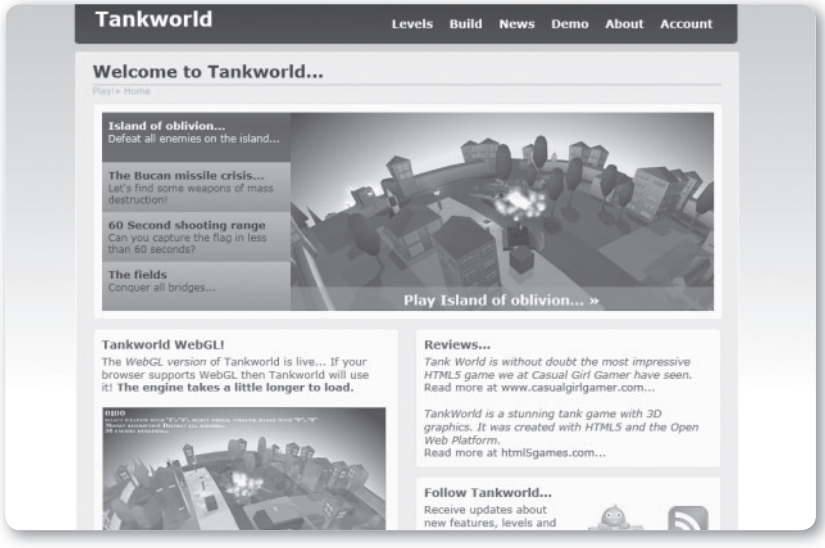


Figura 8. Una nueva generación de juegos ha nacido en la Web con el uso de Canvas y WebGL, como el caso de **`Tank World`** (lo encontramos en el sitio web **`www.playtankworld.com`**).

Para tener más información sobre el elemento Canvas, podemos leer la especificación publicada por el W3C en el sitio que se encuentra en la dirección **www.w3.org/TR/html5/the-canvas-element.html**.

Cabe destacar que la aplicación de esta característica en la Web actual se encuentra en crecimiento. Además, constituye una de las ventajas introducidas por HTML5 que mayores expectativas han creado por la variada gama de posibilidades que incorpora su llegada.

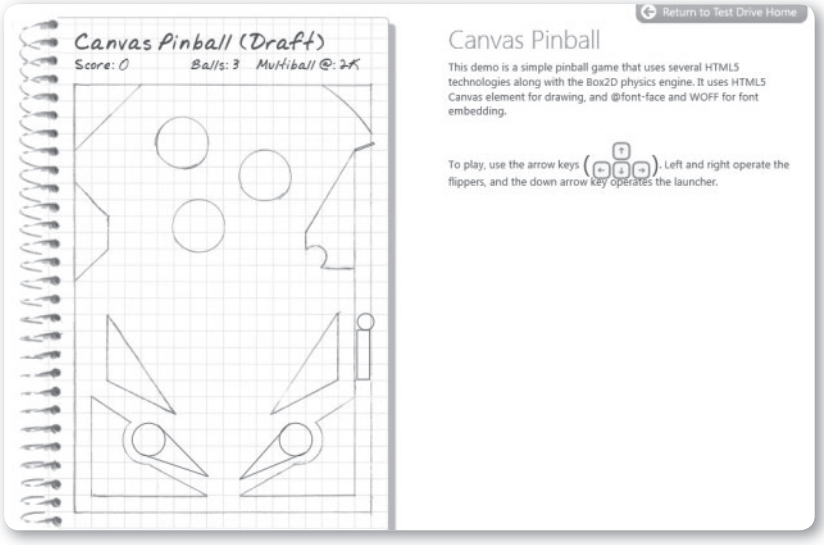


Figura 9. Canvas `Pinball` (**`http://ie.microsoft.com/testdrive/Graphics/CanvasPinball/Default.html`**) es la demo de un juego de Pinball creada por Microsoft utilizando **Canvas**.

SVG

Las siglas **SVG** se refieren al término inglés *Scalable Vector Graphics*, cuya traducción al idioma español podría ser **Gráficos Vectoriales Escalables**. ¿Para qué sirve esta especificación? Es una manera muy eficaz de describir gráficos vectoriales, que además puede trabajar con texto y embeber gráfico de mapa de bits. Para ser más estrictos

LA LIBRERÍA RGRAPH PARA HTML5

RGraph (que encontramos en el sitio web **www.rgraph.net**) es una potente librería que aprovecha las nuevas características introducidas por HTML5 para trabajar con gráficos. Se apoya especialmente en las opciones de **Canvas** y nos permitirá dibujar diferentes tipos de gráficos con gran facilidad, incluyendo gráficos tipo torta, de barras e histogramas, entre otras alternativas disponibles.

en su definición, podemos decir que es un lenguaje de marcado para describir gráficos en dos dimensiones (gráficos 2D con XML).

Recordemos que un gráfico vectorial, a diferencia de uno de mapa de bits, está definido por fórmulas matemáticas que permiten describir figuras geométricas (y también sus características y propiedades).

La ventaja que introduce SVG, estándar del W3C extendido del XML, es que, además de gráficos estáticos, también permite definir formas que pueden moverse, es decir, animaciones.

Figura 10. SVG es estándar del W3C desde el año 2001 podemos saber más sobre él ingresando en **`www.w3.org/Graphics/SVG`**.

Al ser un lenguaje extendido del XML, SVG puede escribirse desde cualquier editor de texto, si es enfocado a código, mejor; pero también existen algunas soluciones que nos ayudan a dibujar de una manera más sencilla para obtener archivos sin tener que escribir el código.

Recordemos que con la etiqueta **`<svg>`**, es posible dibujar aplicando características de este estándar en nuestro documento.

A continuación, veremos un ejemplo del código que necesitamos utilizar para poder dibujar un círculo azul:

```
<svg height="300" width="250" xmlns="http://www.w3.org/2000/svg">
   <circle cx="100" cy="200" r="50" fill="blue" />
</svg>
```

En este ejemplo, a la etiqueta **<svg>**, le aplicamos atributos de alto (**height**) y ancho (**width**), y **xmlns** para definir la dirección URL del estándar empleado. Dentro de la estructura de nuestro SVG, definimos un círculo con la especificación de marcado de SVG (**<circle>**), le especificamos coordenadas de X e Y (**cx** y **cy**), el radio (**r**) y el color de relleno (**fill**).

Tengamos en cuenta que se trata de un ejemplo sencillo; SVG es un formato muy poderoso que puede realizar gran variedad de gráficos y ofrecer soluciones que pueden ser muy eficientes.

Para conocer más sobre la especificación del lenguaje y cómo emplearlo, podemos leer la documentación del W3C sobre la versión 1.1 (**www.w3.org/TR/SVG11/intro.html**) y para la versión Tiny 1.2 (**www.w3.org/TR/SVGTiny12/intro.html**). Existe también un perfil de SVG enfocado a móviles: **www.w3.org/TR/SVGMobile**.

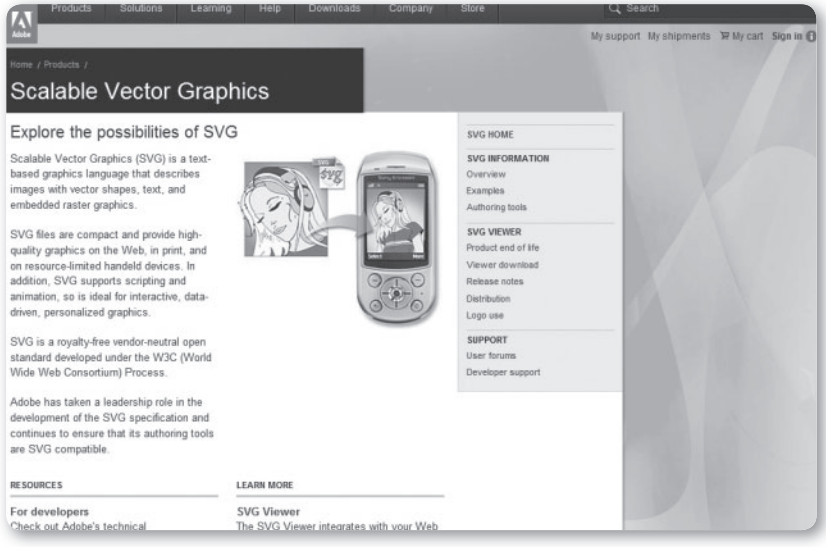


Figura 11. Adobe es una empresa que ha utilizado el formato SVG para potenciar las características que pueden ofrecer sus productos encontraremos más información en **www.adobe.com/svg**.

WebGL (3D)

En el mundo actual, las tecnologías 3D se han introducido en diferentes medios, tanto audiovisuales como multimedia. En la Web y de la mano de **WebGL**, también hace su aparición una característica que incorpora varias novedades en el mundo de la Web 3.0.

Como mencionábamos y veíamos anteriormente, con el uso de WebGL se puede dotar al elemento Canvas de HTML5 de la posibilidad de ofrecer gráficos en 3D. Básicamente, WebGL actúa como una interfaz entre JavaScript y OpenGL ES en su versión 2.0.

Sabemos que WebGL es una característica que puede funcionar en los navegadores que soporten Canvas de HTML5 y así aprovechar la aceleración de las tarjetas de video con OpenGL ES 2.0.

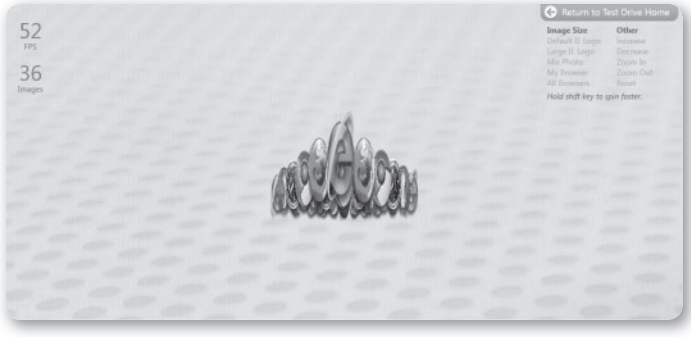


Figura 12. En **`http://ie.microsoft.com/testdrive/Performance/FlyingImages/Default.html`** hay un test FPS.

Para utilizar WebGL con Canvas, al llamar a **getContext**, le indicamos que es **webgl**, en lugar de **2D** como vimos en los ejemplos anteriores. La llamada del contexto aplicada a una variable sería:

```
var variablegl = canvas.getContext('webgl')
```

Al definir el contexto, también podemos utilizar otros parámetros adicionales que son opcionales: **alpha**, **depth**, **stencil**, **antialias**, **premultipliedAlpha** y **preserveDrawingBuffer**.

Veamos un ejemplo donde definimos el canal alpha como falso:

```
var mivariable = canvas.getContext('webgl',{ antialias: false });
```

También podemos definir las dimensiones del Canvas y trabajar con el Viewport, donde se realizará el render resultante.

Figura 13. Junto a las características que se incorporan con Canvas y SVG, WebGL introduce opciones para incorporar 3D y multimedia.

WebGL no es parte de HTML5, solo aprovecha características introducidas para trabajar en su contexto. Podemos leer más sobre su especificación en **www.khronos.org/registry/webgl/specs/1.0**.

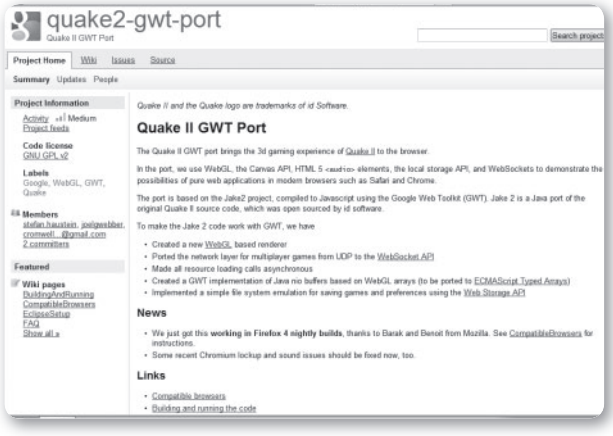


Figura 14. En **`http://code.google.com/p/quake2-gwt-port`** vemos un proyecto que usa WebGL, Canvas y audio HTML5.

WebSockets

Otra de las renovaciones de HTML5 llega con la posibildiad de utilizar una API que permite que nuestras aplicaciones web accedan a **WebSockets**, un protocolo que ofrece características de comunicación bidireccional con la posibilidad de comunicar el cliente con el servidor.

La introducción de esta tecnología permite lograr muy buenos resultados en lo que se refiere tanto a la latencia como al ancho de banda que se puede ahorrar. Esto resulta muy importante como solución para aplicaciones robustas, que requieren alternativas de gran escalabilidad.

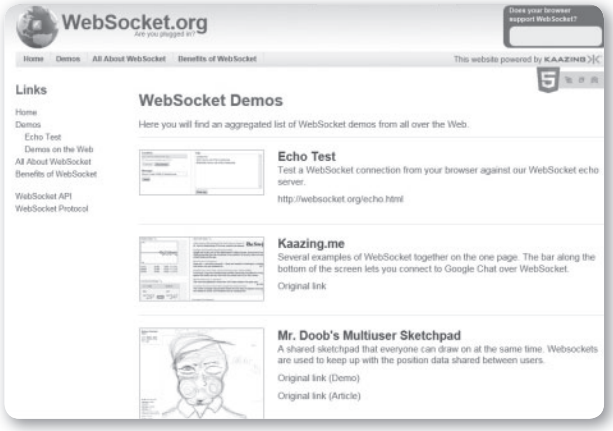


Figura 15. Si ingresamos en el sitio **`http://websocket.org`**, encontraremos información y demos sobre esta especificación.

Definimos el acceso a la interfaz de WebSockets con un constructor, al que podemos pasarle argumentos. El primero de ellos es la URL a la que deseamos realizar la conexión; también puede recibir el protocolo que se empleará, y podremos pasarle el puerto de conexión. Una forma

OPENGL ES

OpenGL es una importante librería gráfica que permite, a través de su API, acceder a la creación de aplicaciones que utilicen gráficos en dos o también en tres dimensiones. **OpenGL ES** se define como una API basada en OpenGL, pero para sistemas embebidos. Su especificación la podemos consultar en el sitio web que encontramos en la dirección **www.khronos.org/opengles**.

sencilla de crear el constructor sería primero asignarlo a una variable en JavaScript, como vemos a continuación:

```
var MiWebSocket = new WebSocket("ws://servidor.prueba.com");
```

Los protocolos típicos para WebSockets son **ws** o **wss**; este último se utiliza para realizar conexiones seguras.

Los atributos soportados por WebSocket son **readyState** y **bufferedAmount**. Con **readyState**, se puede representar el estado de la conexión. El valor **0** indica que la conexión aún no se estableció; el valor **1** se utiliza para indicar que la conexión está activa y se puede realizar la comunicación; con el **2**, se establece que la conexión se está cerrando, y con el **3** se hace referencia a que la conexión ya está cerrada. Con **bufferedAmount**, se determina el número de bytes del texto (en UTF-8) que se pueden enviar mediante **send()**.

SI IMPLEMENTAMOS WEBSOCKETS DEBEREMOS ATENDER ESPECIALMENTE LOS TEMAS DE SEGURIDAD.

WebSockets puede trabajar con los métodos **send()**, que se encarga de recibir como parámetro la información que se envía; y tambén con **close()**, que se utiliza para cerrar.

Es necesario tener en cuenta que WebSockets también utiliza eventos, entre los que encontramos: **onopen** (ocurre cuando el socket se abre), **onmessage** (ocurre cuando el mensaje se recibe), **onerror** (cuando se produce un error) y **onclose** (cuando el socket se cierra).

Con WebSockets, se pueden crear distintos tipos de aplicaciones, desde salas de chat hasta plataformas de juegos multiplayer.

Encontraremos más información sobre la especificación de WebSockets ingresando en **www.w3.org/TR/websockets**.

ACELERACIÓN POR HARDWARE

Debemos recordar que la nueva generación de navegadores de escritorio ha comenzado a incluir, entre sus características, la aceleración mediante hardware, aprovechando las posibilidades que brinda el procesador de video (GPU). En algunos navegadores, esta posibilidad se encuentra habilitada de manera predeterminada, pero, en otros casos, será necesario configurarla en forma manual.

Figura 16. `Atmosphere` (que se encuentra en la dirección web **`http://atmosphere.java.net`**) es un framework que cuenta con soporte para que trabajemos con WebSocket.

Geolocalización

Las posibilidades de localizar, obtener y mostrar la ubicación de un dispositivo, en cualquier lugar del planeta donde se encuentre, es una característica que ha tomado gran relevancia en la Web actual, tanto para dispositivos móviles como también para equipos de escritorio.

Si bien la posibilidad de utilizar geolocalización no es un elemento propiamente dicho de HTML5, sí es una característica que está ligada en su totalidad con este nuevo estándar.

¿Cómo se puede realizar la localización geográfica de un dispositivo? Una de las formas más comunes es mediante un GPS. En los móviles de última generación, es cada vez más común encontrar esta característica, pero no todos los dispositivos cuentan con ella.

Otros métodos están relacionados con la detección por alguna de las características de la red con la cual se está conectando el usuario, por

ejemplo, la dirección IP, a través de la cual se puede contar con información accesible para geolocalizar el elemento. El problema es que no llega a ser tan precisa como el GPS. También podemos usar Wi-Fi (por su MAC) o por RFID (*Radio Frequency IDentification*).

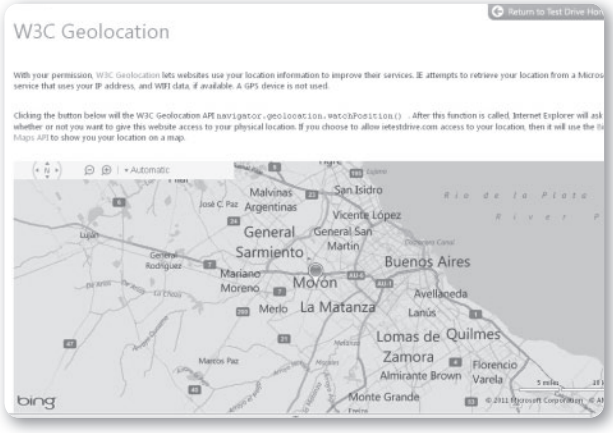


Figura 17. En **`http://ie.microsoft.com/testdrive/HTML5/Geolocation/Default.html`**, vemos un ejemplo de geolocalización.

Geolocation Working Group (**www.w3.org/2008/geolocation**) es el grupo que se encarga de este proyecto, y su misión es brindar una interfaz segura y confiable para poder realizar detecciones del lado cliente, sobre su ubicación geográfica mediante aplicaciones web.

Muchos navegadores modernos han comenzado a soportar esta característica, pero antes de obtener datos de geolocalización del equipo cliente muestran una alerta para prevenir al usuario sobre esto, como medida de seguridad y para salvaguardar la privacidad.

DEV.OPERA

En el sitio web que podemos encontrar en la dirección **http://dev.opera.com**, es posible acceder a una gran cantidad de información muy útil para desarrolladores web, incluyendo artículos relacionados con las últimas tecnologías, acceso a librerías, foros de discusión y varias opciones relevantes más. También debemos tener en cuenta que se puede acceder al denominado **Opera Widgets SDK** adecuado para crear widgets cross-platform y cross-device, que respetan las especificaciones del W3C, presentes en el sitio web que se encuentra en la dirección **www.w3.org/TR/widgets**).

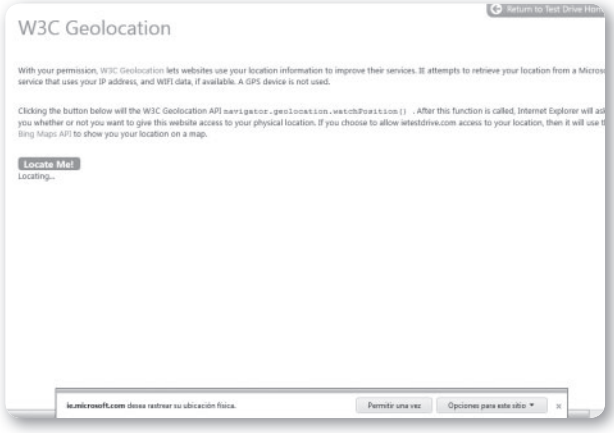


Figura 18. En la especificación de la API de geolocalización, existen apartados relacionados con la privacidad que deben ser respetados.

Mediante el objeto **geolocation**, podemos llegar a detectar si el navegador es compatible con esta característica (por medio de la interfaz **navigator**). Con una estructura condicional de JavaScript, podríamos preguntar si **navigator.geolocation** nos devuelve **true**, para verificar que el navegador es capaz de soportar esta característica.

Tengamos en cuenta que los métodos para trabajar con geolocalización son **getCurrentPosition()**, **watchPosition()** y **clearWatch()**.

El método **getCurrentPosition()** nos brinda la información de posición al realizar el pedido. Entre los parámetros de opciones que puede recibir, el primero está relacionado con la función que se llama cuando la obtención de datos se resuelve de manera correcta; el segundo es para la función en caso de error, y los siguientes son opciones adicionales.

En el caso de **watchPosition()**, se trata de un método que puede "vigilar" en tiempo real el desplazamiento del dispositivo en intervalos de tiempo, lo que nos permite trazar una ruta de movimiento. Debemos tener en cuenta que puede recibir los mismos parámetros que **getCurrentPosition()**.

> LA GEOLOCALIZACIÓN ES UN OPCIÓN CADA VEZ MÁS UTILIZADA EN APLICACIONES WEB.

El método denominado **clearWatch()** nos puede servir para limpiar un seguimiento, es decir, se encarga de realizar la detención de **watchPosition()**. Para lograr esto, le deberíamos pasar como parámetro, a **clearWatch()**, el identificador de **watchPosition()**.

Las propiedades conocidas como **timestamp** y **coords** pueden ser enviadas por **getCurrentPosition()** o también por **watchPosition()** a los objetos que definamos para este fin.

Como el lector podrá imaginar, **timestamp** es un dato de tiempo (en milisegundos), que expresa cuándo fue obtenida la ubicación.

En el caso de **coords**, encontraremos las coordenadas: **latitude** (latitud, en grados decimales), **longitude** (longitud, en grados decimales), **altitude** (altitud, en metros), **accuracy** (precisión de la detección de latitud y longitud, en metros), **altitudeAccuracy** (precisión de la detección de altitud, en metros), **heading** (dirección de movimiento, en grados decimales) y **speed** (velocidad referida al movimiento, en metros por segundo). Las últimas dos opciones, tanto **heading** como **speed**, funcionan solamente con **watchPosition()**, ya que están relacionadas con objetos en movimiento.

Para saber más sobre la especificación del W3C sobre la API de geolocalización ingresamos en **www.w3.org/TR/geolocation-API**.

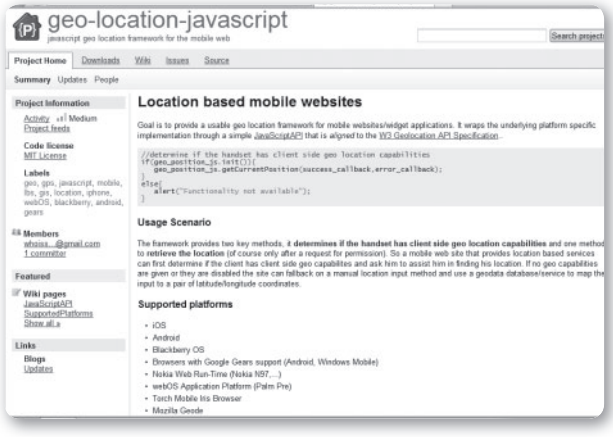


Figura 19. En geolocalización para móviles, existe una excelente herramienta denominada **geo-location-javascript**.

Detección de compatibilidad

Con lo visto anteriormente, ya tenemos todos los detalles para crear nuestros códigos y emplearlos en nuestros proyectos. Podríamos crear una función para realizar la detección de la posición del usuario y mostrarla en un mapa o escribir los datos en pantalla. También podríamos pasar los parámetros de latitud o longitud para geolocalizar un punto que queramos mostrar en un sitio web, por

ejemplo, la ubicación de un comercio, un restaurante o cualquier punto de referencia que deseemos.

Hemos visto que contamos con **navigator.geolocation** para ayudarnos con la detección, pero también es posible trabajar con un framework, como el caso de Modernizr, mediante la propiedad **Modernizr.geolocation** (**www.modernizr.com/docs/#geolocation**).

La enorme variedad de sistemas de mapas que existen en la actualidad nos brinda una alternativa de gran utilidad para cubrir nuestras necesidades, si precisamos una opción diferente para nuestro desarrollo. A continuación, veremos la alternativa que nos ofrece Google Maps.

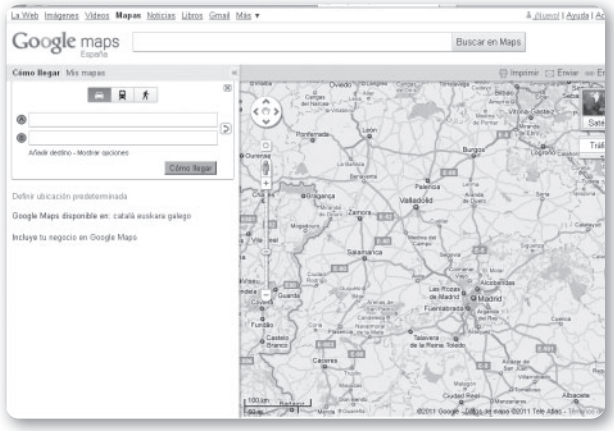


Figura 20. Las opciones que brinda **Google Maps** son variadas. Por ejemplo, es posible buscar ayuda para llegar a una ubicación.

Geolocalización con Google Maps

Google Maps (**http://maps.google.com**) es uno de los servicios de mapas más populares en Internet y uno de los servicios gratuitos que nos brinda el gigante de los buscadores. Es compatible con los

LAS APIS DE GOOGLE MAPS

Si ingresamos en **Google Maps API Family** (**http://code.google.com/intl/es-ES/apis/maps/index.html**), podremos acceder a las diversas alternativas de la API de Maps de Google para integrarlas a nuestros proyectos web: Maps JavaScript API, Maps API for Flash (mediante ActionScript API), Google Earth API, Static Maps API, Servicios web (empleando JSON o XML) y Maps Data API.

navegadores de escritorio y también con varias plataformas móviles, ya que la conexión se trabaja mediante AJAX.

Como desarrolladores, debemos recordar que es posible aprovechar sus características para ofrecer un mapa con la localización de un elemento en cualquier lugar del globo terráqueo.

Para que podamos acceder a trabajar con la API de Google Maps debemos utilizar el objeto **google.maps.Map**.

Si buscamos hacer que en el mapa se muestre una ubicación fija de un lugar que deseamos mostrar, podemos pasarle los parámetros instanciando **google.maps.LatLng**. Veamos un ejemplo a continuación que nos sitúa en la Ciudad de Buenos Aires.

Primero vamos a incluir la llamada a la API de Google Maps en la cabecera, para lo cual copiamos el siguiente código:

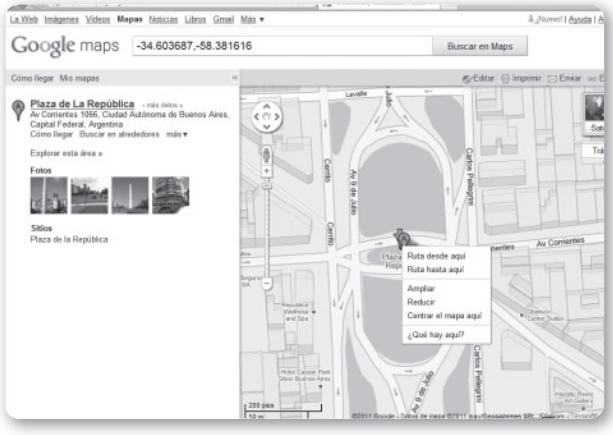


Figura 21. Para conocer las coordenadas vamos a **http://maps.google.com**, hacemos clic derecho y elegimos la opción **¿Qué hay aquí?**

```
<script type="text/javascript" src="http://maps.google.com/maps/api/
js?sensor=false"></script>
```

El parámetro que podemos cambiar según nuestra necesidad es el valor de **sensor**. Debemos tener en cuenta que se puede emplear **true** si deseamos utilizar características del sensor del dispositivo del usuario (por ejemplo, un dispositivo GPS). Si no requerimos esta característica, podemos indicar **false**, tal como se especifica en el ejemplo

El script que mostramos a continuación se encarga de inicializar y proveer los datos necesarios para la API:

```
<script type="text/javascript">
  function inicializar_mapa() {
    var LatBaires = new google.maps.LatLng(-34.603647,-58.381611);
    var Opciones = {
      zoom: 16,
      center: LatBaires,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById("mapa"), Opciones);
  }
</script>
```

Luego, tengamos presente que en el cuerpo de nuestro documento debemos cargar la función, que en este caso la hemos llamado **inicializar_mapa**, de la siguiente forma:

```
<body onload="inicializar_mapa()">
```

A continuación, simplemente deberemos aplicarle la **id** que pasamos a **document.getElementById** a un elemento HTML, por ejemplo a un ** **, y establecer las dimensiones con las que se mostrará el elemento y su contenido (el mapa) en la pantalla. Esto podemos hacerlo mediante las propiedades **width** y **height** de CSS.

Figura 22. El resultado de la geolocalización de Buenos Aires, en este caso la Plaza de la República, lo veremos como se ilustra en esta figura.

Drag & Drop

Las opciones de arrastrar y soltar (**Drag & Drop**) son muy utilizadas en el diseño y desarrollo web actual, ya que le brindan al usuario una manera más práctica de interactuar con la interfaz.

Gracias a la posibilidad de acceder a la API de JavaScript, a partir de HTML5 encontramos la opción de contar con Drag & Drop nativo en los navegadores capaces de soportar esta característica.

Figura 23. Tengamos en cuenta que en la actualidad, muchas necesidades de Drag & Drop aún se manejan empleando técnicas relacionadas con AJAX, como por ejemplo, el Dashboard de WordPress.

¿QUÉ ES GOOGLE LATITUDE?

Google Latitude (que se encuentra en el sitio web con la dirección **www.google.com/latitude**) es un interesante y útil servicio de geolocalización, cuyo lanzamiento ocurrió en el año 2009. La idea de este servicio de Google es poder tener a todos nuestros amigos localizados en un mapa y, también, tener la posibilidad de compartir nuestra propia ubicación, si lo deseamos.

Seguramente, el lector recordará el atributo **draggable** introducido en HTML5. Una condición para que un elemento pueda ser arrastrado es que tenga este atributo colocado en **true** (desde el marcado o por medio de la manipulación del DOM). Luego, lo que necesitaremos, además del objeto para arrastrar, es otro que actúe como posible destino.

Los eventos relacionados con Drag & Drop son:

- **dragstart**: es el comienzo de la operación de arrastrar.
- **drag**: ocurre cuando se está arrastrando el elemento con el mouse.
- **dragenter**: sin ser soltado, el elemento se coloca dentro del destino.
- **dragover**: ocurre cuando el elemento es pasado por encima del destino.
- **dragleave**: ocurre cuando el elemento que estamos arastrando es soltado fuera del elemento receptor.
- **drop**: tengamos en cuenta que se produce cuando el elemento arrastrado se suelta sobre el destino correspondiente.
- **dragend**: ocurre cuando el evento que estamos arrastrando se concluye, finalizando la acción.

Además de arrastrar elementos que forman parte del DOM, también es importante señalar que es posible transferir la información que contiene el elemento arrastrado a su destino, tanto sea en el mismo documento como desde otra ventana. Esto se puede realizar empleando el objeto **dataTransfer**, que puede trabajar con los métodos **setData** (nos permite acceder a setear la información) y **getData** (se encarga de obtener la información correspondiente). Entre el tipo de información que podemos manejar se encuentran las opciones de texto, links, HTML o XML, imágenes y también archivos, entre otras interesantes opciones. A continuación, en el código que mostramos, veremos un ejemplo con el tipo texto trabajando con **setData**:

```
event.dataTransfer.setData("text/plain", "Texto que se va a arrastrar")
```

Otra característica interesante es la que nos brinda **setDragImage**, un método que nos permite establecer qué imagen se mostrará cuando estamos arrastrando. También es posible establecer efectos de arrastrado, tanto para el elemento que se va a arrastrar, como para el que lo puede recibir; estas propiedades son **effectAllowed** y **dropEffect**, ambas trabajan con el objeto **dataTransfer**.

La propiedad **effectAllowed** retorna los tipos permitidos del elemento que será arrastrado. Los posibles valores son **none**, **copy**, **copyLink**, **copyMove**, **link**, **linkMove**, **move**, **all** o **uninitialized**. En el caso de **dropEffect** para el elemento destino, puede trabajar con **none**, **copy**, **link** o **move**.

Encontraremos más información sobre Drag & Drop en la documentación que nos ofrece el W3C en el sitio que se encuentra en la dirección **www.w3.org/TR/html5/dnd.html**.

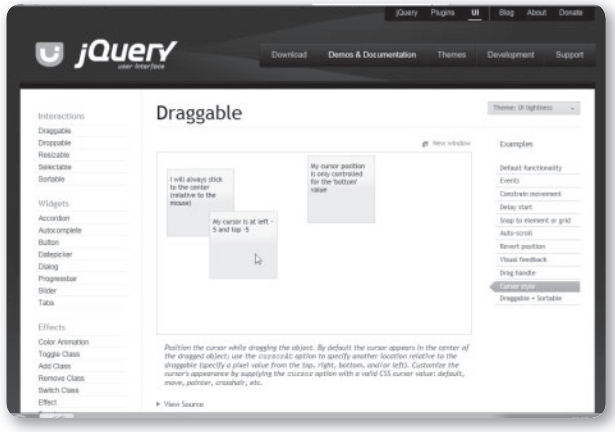


Figura 24. En el sitio **`http://jqueryui.com/demos/draggable`** hay una solución para navegadores que aún no soportan HTML5.

Web Storage

Con **Web Storage**, podremos almacenar información en el equipo cliente y también datos de sesiones. Esta información cuenta con pares de claves y valores (clave=valor) que se guardan en el equipo cliente para que puedan ser accedidos posteriormente por nuestra aplicación web.

Web Storage llega para suplir algunas de las falencias conocidas con el uso de cookies en los desarrollos. Cabe decir que el navegador no elimina por defecto la información almacenada, pero el usuario sí puede elegir hacerlo si lo desea.

Para utilizar esta característica, en primer lugar necesitamos introducir unas líneas de código que trabajan con la librería Modernizr (que encontramos en la dirección **www.modernizr.com/docs**) para saber si encontramos compatibilidad por parte del navegador.

```
if (Modernizr.localstorage){
  // Aquí va el código resuelto con Web Storage.
} else {
  // Aquí va el código alternativo.
}
```

En este caso, estamos verificando **Local Storage** con Modernizr; si deseáramos verificar **Session Storage**, deberíamos crear la estructura empleando **Modernizr.sessionstorage**. Es importante tener en cuenta que la diferencia entre Local Storage y Session Storage reside en la persistencia de los datos, ya que el último solo se enfoca en los datos de la sesión.

Almacenamiento local en nuestros desarrollos

Es necesario tener en cuenta que si decidimos implementar almacenamiento en el equipo local en nuestro desarrollo, deberemos conocer algunas de sus características principales.

El objeto **sessionStorage** se utiliza para almacenar y manejar datos de una sesión; es decir que, cuando el navegador o la pestaña se cierran, o si se sale de la aplicación cerrando la sesión, esta información dejará de estar disponible.

En el caso de **localStorage**, se trata de un objeto que está pensado para que dure más allá de la sesión, ya que se puede eliminar o bien desde el Script o ante la acción específica del usuario.

Es importante tener en cuenta que, de igual modo que ocurre con el uso de cookies, por esta razón sabemos que no es recomendable almacenar información sensible con Web Storage.

EL CAMINO DE LA ESTANDARIZACIÓN

Como sabemos, HTML5 es un lenguaje aún en proceso de estandarización. Si deseamos aprender más sobre los diferentes niveles y pasos que llevan adelante los grupos de trabajo del W3C hasta que una tecnología web llega a ser estándar, podemos leer el documento que se encuentra en la dirección **www.w3.org/2005/10/Process-20051014/tr**.

Los métodos que se utilizan junto a Web Storage son:

- **setItem()**: se trata de un método que permite establecer una entrada que recibirá la clave y su valor, que serán pasados como los parámetros para este método. Veamos un ejemplo:

```
sessionStorage.setItem('Nombre_Clave','Valor_Clave');
```

- **getItem()**: este método permite recuperar el valor de una clave pasada como parámetro. Por ejemplo, para obtener el valor almacenado en la clave **usuario**. Para una sesión escribimos:

```
sessionStorage.getItem('usuario');
```

- **clear()**: limpia las claves y su valor. Para una sesión se puede escribir:

```
sessionStorage.clear()
```

- **removeItem()**: le pasamos una clave como parámetro y remueve la clave y su valor, del objeto **sessionStorage**. Por ejemplo, para remover la clave **apellido** y su valor, escribimos

```
sessionStorage.removeItem('apellido');
```

- **key()**: obtiene el valor de una clave (enésima) y recibe como parámetro un índice (n). Aplicando un código JavaScript para ciclar, lograríamos recuperar todas las claves almacenadas que tuviéramos disponibles.

En los ejemplos que vimos antes, en lugar de **sessionStorage**, también podemos trabajar con **localStorage**, según corresponda.

Un aspecto importante cuando trabajamos con Web Storage es que los eventos no se disparan necesariamente cuando se invoca un método relacionado; un evento se produce solo cuando algo cambia efectivamente en el almacenamiento. Los atributos relacionados son:

- **key**: se trata del nombre de la clave afectada (que pudo haber sido agregada, modificada o también borrada).
- **oldValue**: el valor previo que tenía el ítem.
- **newValue**: el nuevo valor que tiene ahora el ítem.
- **url**: la URL que llama al método que modifica el almacenamiento.
- **storageArea**: el objeto de almacenamiento.

Figura 25. Podemos observar un ejemplo de Web Storage ingresando en el sitio **`http://html5demos.com/storage`**.

Para conocer el estado actual del documento publicado por el W3C sobre Web Storage, podemos ingresar en el sitio web que se encuentra en la dirección **www.w3.org/TR/webstorage**.

Web Messaging

Es interesante tener en cuenta que, por algunas razones de seguridad, determinadas características de comunicación entre documentos que se encuentran alojados en diferentes sitios están bloqueadas. La idea de esta solución, mediante el uso de **Cross-document messaging**, es que pueda realizarse una comunicación de una manera efectiva y segura para transmitir los mensajes necesarios.

Si se emplea el método denominado **`postMessage()`**, es posible que accedamos a establecer una comunicación en la que el origen puede recibir datos sobre el dominio, y de esta forma posibilita realizar la verificación si es el que se espera. Es importante tener mucha precaución al escribir el código que utilice esta característica ya que, si no se hace de manera correcta, podemos dejar abierta la puerta a un problema de seguridad importante. Este método trabaja con el objeto de la ventana, y los parámetros que puede recibir son: el mensaje, el origen y el puerto. Con frecuencia, estos pueden integrarse en el proyecto mediante un **`<iframe>`**.

Figura 26. Podemos ver un ejemplo de mensajes **`Cross Domain`** en el sitio **`http://html5demos.com`**, seleccionando postMessage.

Si necesitamos proceder a crear un desarrollo que tenga mensajes de dos vías, podemos utilizar **MessageChannel()**.

Tanto Cross-document messaging como Channel messaging utilizan el evento **message**. Con el método **initMessageEvent()**, podrenos inicializar el evento. Los atributos con los que podemos trabajar son:

- **data**: la información que será enviada en el mensaje.
- **origin**: el origen del documento que envía el mensaje.
- **lastEventId**: íd de evento del evento origen.
- **source**: este atributo se encarga de representar el WindowProxy de la ventana origen en Cross-document messaging.
- **ports**: es el array de los puertos de los mensajes que se envían, tanto para Cross-document messaging como para Channel messaging.

Para saber más sobre esta especificación y conocer los recaudos de seguridad que hay que tener en cuenta es recomendable leer la documentación provista en **www.w3.org/TR/webmessaging**.

MANTENERSE INFORMADO Y ACTUALIZADO

HTML5 es la versión que llegará a ser estándar en el año 2014. Hasta ese momento, varias de sus características se mantendrán en evolución. Por este motivo, es recomendable mantenernos informados y actualizados con los datos disponible en **www.w3.org/TR/html5** y también con las novedades publicadas en el **W3C Editor's Draft** (**http://dev.w3.org/html5/spec/Overview.html**).

Server-Sent Events

Con la especificación de Server-Sent Events, se define una API para abrir una conexión mediante HTTP y permitir la realización de streaming a través de este protocolo de manera unidireccional.

La idea es que se permita al cliente recibir notificaciones abriendo un flujo cuando se crea un evento que recibe mensajes. El evento **onmessage** actúa como manejador y pide la nueva información desde el cliente, estableciendo una conexión asincrónica con el servidor.

Figura 27. Web Sockets y **Server-Sent Events** introducen características eficientes para interacción en juegos y aplicaciones.

Según indica el W3C, para trabajar con esta API es necesario instanciar el objeto **EventSource** y registrar el evento que actuará como escucha o listener. El código de ejemplo es el siguiente:

```
var source = new EventSource('updates.cgi');
source.onmessage = function (event) {
  alert(event.data);
};
```

Aquí vemos a **EventSource** actuando como constructor y, claro, el recurso ejemplificado por el W3C (**updates.cgi**) deberá ser reemplazado por el que nosotros designemos en nuestro desarrollo. Es preciso tener en cuenta es que dicho recurso deberá emitir mensajes empleando el tipo MIME **text/event-stream**. Veamos un ejemplo:

```
data: primer texto\n
data: segundo texto\n\n
```

Debemos tener presente que el uso de JSON puede servirnos como una opción para ayudarnos a interactuar de manera asincrónica con la información. Esta interfaz de envío de mensajes desde el servidor también puede trabajar con el atributo denominado **readyState**, que representa el estado de la conexión y puede tener como valores: **0** (conectando), **1** (abierta) o **2** (cerrada).

Los manejadores de evento con los que contamos son **onopen**, **onmessage** y **onerror**. Para cerrar la conexión, utilizamos el método **close()**.

Podemos encontrar más información sobre Server-Sent Events en el artículo publicado por el W3C en el sitio web que se encuentra en la dirección **www.w3.org/TR/eventsource**.

Figura 28. **The HTML5 Test** (**http://html5test.com**) es un test para conocer la compatibilidad del navegador con HTML5.

Web Workers

Con **Web Workers**, se define una API que nos brinda la posibilidad de correr scripts de fondo, por detrás de la interfaz de usuario; de esta forma, podremos evitar que se congestione su funcionamiento en primer plano. Para nuestros proyectos, por lo general, estos códigos estarán escritos en el lenguaje JavaScript.

Existen muchos usos posibles de Web Worker; por ejemplo, se puede emplear para realizar actualizaciones en una base de datos del lado cliente, o cualquier operación que provoque demoras importantes en la aplicación que está corriendo en primer plano.

Es importante verificar si el cliente que accede cuenta con un navegador compatible. Una vez más, podemos recurrir a Modernizr (**www.modernizr.com/docs**) para que nos ayude con este fin, utilizando la propiedad **Modernizr.webworkers** para que realicemos la comprobación del soporte al objeto **window.Worker**.

Luego, para utilizarlo hay que crear una instancia del objeto. A continuación, especificamos qué ocurre cuando el mensaje es enviado, armando la parte de comunicación entre los actores y, por último, podemos crear alguna línea que trabaje con los posibles errores.

En el siguiente ejemplo es posible observar la creación del objeto y la inicialización con el método **postMessage()**:

```
var worker = new Worker('trabajo.js');
worker.postMessage();
```

Por supuesto, debemos crear el código del .JS externo con el contenido que deseemos que corra como aplicación en segundo plano.

Luego, tenemos que definir un manejador de eventos con **onmessage** del evento **message**. Vale mencionar que los mensajes que pasan son copiados para estar disponibles en ambas locaciones.

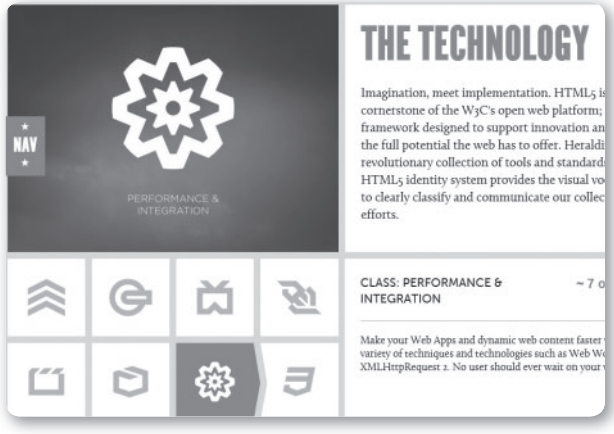


Figura 29. Con WebWorkers, podremos lograr que nuestras aplicaciones se comuniquen y funcionen con una mejor performance.

Es importante aclarar que el Worker no tiene acceso al DOM ni a los objetos denominados **window**, **document**, ni **parent**.

Más información sobre esta especificación, en **www.w3.org/TR/2009/WD-workers-20090423** y en el W3C Editor's Draft: **http://dev.w3.org/html5/workers**.

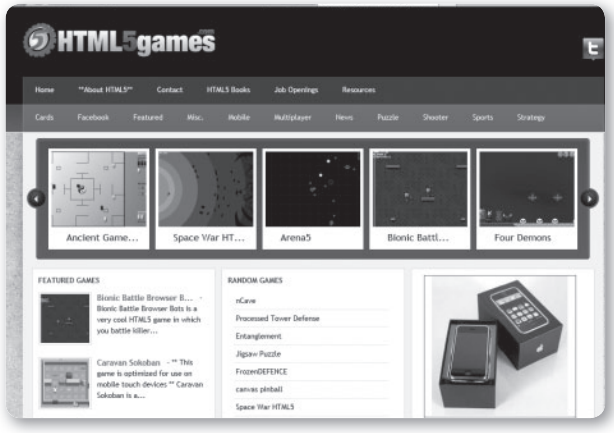


Figura 30. El sitio **`http://html5games.com`** nos brinda una interesante galería de juegos que aprovecha las características de HTML5.

Application Cache API

La llegada de HTML5 plantea una solución muy interesante para la creación de aplicaciones, con las que se puede trabajar estando online u offline. Claro que lo que hagamos offline luego podrá ser sincronizado o enviado al servidor cuando nos conectemos nuevamente a la aplicación web.

IAN HICKSON

Nacido en Ginebra, Suiza, **Ian Hickson** es editor de la especificación de HTML5, y autor de Acid2 y Acid3. En su importante currículum, se destaca su labor en Netscape, Opera Software y Google (donde trabaja actualmente). Además de su trabajo vinculado con HTML5, también ha participado en otros proyectos muy importantes y es uno de los editores de la especificación de CSS 2.1.

Modernizr nos provee una forma sencilla de detectar si el navegador tiene la capacidad de trabajar con Application Cache (**www.modernizr.com/docs/#applicationcache**); solo debemos hacer una comprobación mediante un script:

```
if (Modernizr.applicationcache){
  // El navegador es compatible. Aquí va el código del script cuando se presenta compatibilidad.
} else {
  // El navegador no es compatible. Se muestra un mensaje indicándole al usuario esto.
}
```

Si deseamos hacer referencia a Aplication Cache mediante el DOM sin utilizar Modernizr, deberíamos recurrir a **window.applicationCache**.

Tengamos en cuenta que gracias al empleo de la propiedad **navigator.onLine**, tenemos la posibilidad de trabajar con los valores **true** (online) o **false** (offline) y, de esta manera, contar con la información que nos interesa sobre el modo en que está actuando el navegador.

Es importante recordar que, entre los eventos soportados por **<body>** a partir de la versión 5 de HTML, existen dos que nos serán muy útiles para este tipo de necesidades: en primer lugar tenemos **onoffline** (ocurre cuando el navegador pasa al modo offline) y en segundo, **ononline** (se produce cuando el navegador pasa a estar online). Para ambas situaciones, podemos escribir scripts que corran cuando ocurran estos eventos.

> LAS APLICACIONES WEB QUE PUEDEN TRABAJAR OFFLINE COMIENZAN A MARCAR EL FUTURO.

Cache Manifest

Con **Cache Manifest**, podemos establecer los recursos de nuestra aplicación web que deseamos que puedan trabajar offline. En primer lugar debemos crear el archivo, que es muy simple; solo debemos recordar incluir todas los recursos que necesitamos que funcionen offline. A continuación, veamos un ejemplo (podemos indicar el nombre del recurso o su dirección completa):

```
CACHE MANIFEST
# v1
principal.html
logo.jpg
script.js
```

Posteriormente, debemos agregar la siguiente línea en el **.htaccess** del servidor donde esté alojada la aplicación:

```
addType text/cache-manifest manifest
```

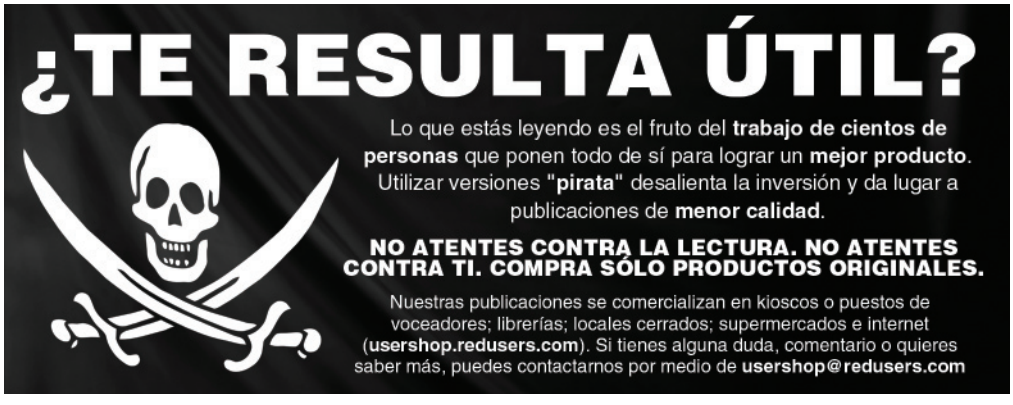
Luego, debemos aplicarle el atributo **manifest** a la etiqueta **<html>** del documento afectado. Veamos el ejemplo a continuación:

```
<html manifest="http://www.misitio.com/cache-manifest">
```

Dicha declaración también se puede realizar de la manera que se ejemplifica en el siguiente código:

```
<html manifest="site.manifest">
```

Los archivos detallados en el listado del Cache Manifest se cargarán en el equipo local cuando el usuario acceda. A partir de allí, podrá usarlos de manera offline si lo desea. Para más información sobre la creación de aplicaciones Offline, podemos leer el documento del W3C **Offline Web Applications** (**www.w3.org/TR/offline-webapps**)



History API

Junto a HTML5, se introduce una interesante API que nos entrega la posibilidad de manejar la información del historial. Esta característica nos posibilita agregar entradas y también trabajar con eventos cuando ocurran cambios en el historial del navegador.

Para efectuar nuestra labor sobre el historial, contamos con el objeto **window.history**, sobre el cual podremos trabajar con algunos atributos y métodos que veremos a continuación:

- **length**: se trata de un atributo que devuelve el número de entradas dentro del conjunto del historial correspondiente.
- **state**: atributo que devuelve el estado actual del objeto.
- **go**: se trata del método que permite avanzar o retroceder, en el historial, la cantidad de pasos indicados.
- **back**: este método es el que nos permite retroceder un paso en el historial si existe, al menos, una página previa.
- **forward**: método que se encraga de permitirnos avanzar un paso en el historial si existe al menos una página siguiente
- **pushState**: se trata del método que permite ingresar una información dada en una entrada de la sesión. Recibe como parámetros la información, el título y la URL.
- **replaceState**: se trata del método para reemplazar la entrada actual en la sesión del historial por la información dada.

RESUMEN

En el capítulo final de este libro, hemos conocido características avanzadas de HTML5 y su interacción con las APIs de JavaScript y DOM. Aprendimos cómo se utiliza el elemento Canvas, qué ventajas nos ofrece SVG y las novedades que introduce WebGL para la producción de contenidos 3D en la Web. Conocimos la importancia de WebSockets y aprendimos a utilizar las características de geolocalización. Vimos, en detalle, las opciones que nos ofrecen Web Storage, Web Messaging, Server-Sent Events y Web Workers. Analizamos los aspectos que hay que tener en cuenta para construir una aplicación web que funcione offline y, para finalizar, vimos algunas APIs y tecnologías vinculadas.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué ofrece el elemento **Canvas** y con qué etiqueta se define?
2. Mencione las ventajas de trabajar con SVG.
3. ¿Qué es **WebGL** y **OpenGL**?
4. ¿Cuáles son las ventajas que brinda utilizar **Web Sockets** con HTML5?
5. Indique cuáles son los eventos relacionados con **Drag & Drop**.
6. Explique el concepto de geolocalización y describa las características de su API.
7. ¿Qué son los **Server-Sent Events**?
8. ¿Qué beneficios se introducen con la posibilidad de utilizar WebWorkers con HTML5?
9. ¿Qué ventajas brinda la posibilidad de crear aplicaciones offline empleando las características que introduce HTML5?
10. Explique qué posibilidades introduce el uso de **API History**.

EJERCICIOS PRÁCTICOS

1. Realice el dibujo de un trapecio y de un hexágono, llenándolos ambos con color marrón por medio del elemento Canvas.
2. Cree una forma geométrica (un círculo) utilizando la especificación de SVG.
3. Escriba el código para almacenamiento local de una aplicación web con HTML5.
4. Muestre su ciudad en un mapa utilizando las características de geolocalización de HTML5 y, luego, realice el ejemplo con Google Maps.
5. Prepare el código del archivo Cache Manifest de una aplicación web que haya desarrollado para que funcione offline.

Servicios al lector

En esta sección nos encargaremos de presentar un útil índice temático para que podamos encontrar en forma sencilla los términos que necesitamos. Además podremos ver una interesante selección de sitios y programas que se encuentran relacionados con el contenido de esta obra.

- Índice temático.....304
- Sitios web relacionados.....307
- Programas relacionados.....311

Indice temático

A

ARPANET 14
Accept 215
Accept-charset 214
Aceleración por hardware 280
Action 213
AJAX 15, 34
Almacenamiento local 291
Animación 182
API de selectores 264
Aplicaciones web móviles 258
Atributos HTML 64
Atributos HTML5 91
Atributos no soportados 87
Atributos soportados 108
Audio 191
Audio.js 200
Automatic 203
Autoplay 192

B

Banda ancha 24
Blubrry 205
Bordes 164

C

Cabecera 102
Cache manifiest 299
Canvas 266
Captcha 225
Cascada 136
Códecs de audio 192
Códecs de video 195
Color 168
Compatibilidad 119
Controls 192
CSS 26
CSS3 30

D

Dailymotion 194
Device 208
Dispositivos multimedia 208
DOM 37
Drag & drop 288

E

Elemento track 197
Elementos HTML4 68
EmailMe 220
Enlaces 123
Entidades HTML 70
Errores de código 93
Especificidad 136
Estandarización 291
Estilos CSS 128
Estructura de páginas 113
Estructura semántica 111
Etiqueta <button> 222
Etiqueta <datalist> 226
Etiqueta <form> 213
Etiqueta <input> 215, 217
Etiqueta <label> 218
Etiqueta <output> 227
Etiqueta <source> 198
Etiqueta <textarea> 219
Etiqueta <video> 194
Etiquetas 102
Eventos HTML 66

F

Facebook 17
Figure 233
Firebug 58
Flash video 197
Fondos 167
For 218

F

Form ... 213
Form check ... 224
Formulario de contacto ... 232
Formularios ... 212
Frameworks ... 40

G

Gecko ... 52
Geolocalización ... 16, 281
Get ... 213
Google Chrome ... 45
Google latitude ... 288
Google Maps ... 285
Gradient ... 169
Gury ... 270

H

H.264 ... 196
H5Validate ... 239
Herencia ... 134
History API ... 300
Hot scripts ... 217
HTML ... 20, 21, 64
HTML5 ... 25, 89
HTML Media Capture ... 208

I

Indentado ... 95
Input ... 227
Inspección de código ... 57
Inteligencia artificial ... 17
Intenet Explorer 9 ... 43
Interactividad ... 186
Internet Explorer ... 42

J

JavaScript ... 32
Jquery ... 224
JPlayer ... 201
JSON ... 38
JSValidate ... 225
JSValidate.com ... 224

L

LAMP ... 237
Lenguajes de etiquetas ... 18
Linecap ... 269
Linewidth ... 268
Loop ... 192
Lynx ... 76

M

Mapa de sitio ... 103
Method ... 213
Microdatos ... 120
Miro Video Converter ... 201
Modernizr ... 60, 207
Mooplay ... 207
Motores de renderizado ... 51
Mozilla Firefox ... 44
MP3 ... 192
Multicolumna ... 159
Multimedia ... 186
Multiple ... 228

N

Name ... 216
Navegabilidad ... 109
Navegadores ... 41
Navegadores para móviles ... 245
Notepad++ ... 55
Number ... 230

O

Ogg Vorbis ... 193
OpenGL ... 279
Opera ... 48
Output ... 227

P

Pagerank ... 122
Palabras clave ... 143
Paleta de colores ... 143
Password ... 217
PHP classes ... 219
Post ... 214

P

Preload 192
Prototype 233
Pseudoclases 151
Pseudoelementos 151

R

Radio 217
Radio button 216
Range 231
Realidad aumentada 16
ReCaptcha 225
Reglas en CSS 131
Rel 108
Reset 217
Rgraph 274
RIA 39
Rows 219
RSS 114

S

Safari 47
Search 231
Selectores 129
Selectores avanzados 130, 150
Semántica 15
Semántica 100, 101
SEO 122
SGML 19
Shortland 133
Size 216
Soluciones multimedia 198
Sombra 156
Sprites 154
SRC 192
Streaming 205
Subcreator 198
SublimeVideo 195
Submit 217
SVG 274
SVGMobile 276

T

Target 215
Test de compatibilidad 59
Text 217
Tel 231
Theora 195
Tim Berners-Lee 18
Time 230
Track 197
Transformación 176
Transición 180
Type 215

U

UML 116
Unidades de medida 153
Usabilidad 214
URL 231

V

Value 216
Video 193
VideoJS 206
Vimeo 190

W

W3C 18, 91
WAMP 237
Week 230
Web 2.0 14
Web 3.0 15
WebM 196
Web messaging 293
Web móvil 242
Web storage 290
Web workers 296
WebGL 277
Webkit 52
WebSockets 279
Width 65
World Wide Web 14, 17
WYSIWIG 53

Sitios Web relacionados

WHATWG ● http://www.whatwg.org

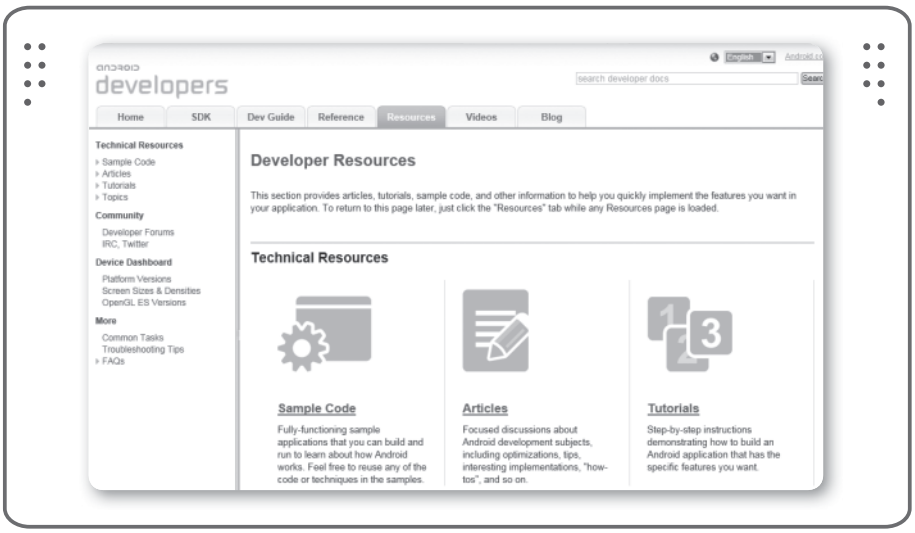
En el sitio www.whatwg.org (Web Hypertext Application Technology Working Group) podremos encontrar todas las novedades publicadas sobre la evolución del estándar HTML. Este grupo trabaja sobre HTML5 y Web Workers, entre otros estándares.

HTML GOODIES ● http://www.htmlgoodies.com/html5

El sitio HTML Goodies nos ofrece una interesante sección especial sobre HTML5 (se encuentra en la dirección www.htmlgoodies.com/html5). Allí nos ofrecen una serie de recursos, artículos y también algunos ejemplos prácticos de la nueva versión del lenguaje HTML.

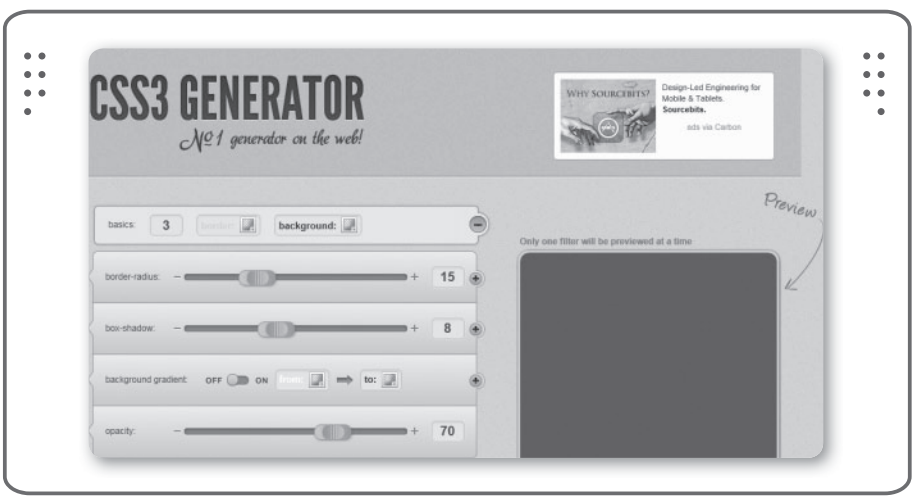
ANDROID DEVELOPER ● http://developer.android.com

El desarrollo para móviles es fundamental en esta etapa de la Web. Android es una de las plataformas más utilizadas del mundo. En Android Developer Resources (http://developer.android.com/resources/index.html) podremos hallar una amplia variedad de recursos para desarrolladores.



CSS3 ● http://www.css3.me

El sitio www.css3.me nos ofrece una manera práctica e interactiva de generar código CSS3. Sin necesidad de escribir una sola línea, podremos crear diversos sombreados y bordes redondeados, o también trabajar sobre la opacidad de un elemento.



PAGESPEED ● http://pagespeed.googlelabs.com

Page Speed de Google Labs (http://pagespeed.googlelabs.com) es una práctica e interesante herramienta online que se encarga de realizar un análisis completo sobre nuestro sitio web, de esta forma nos ayuda a conocer su performance. Sencilla y fácil de usar.

BING WEBMASTER ● http://www.bing.com/toolbox/webmaster

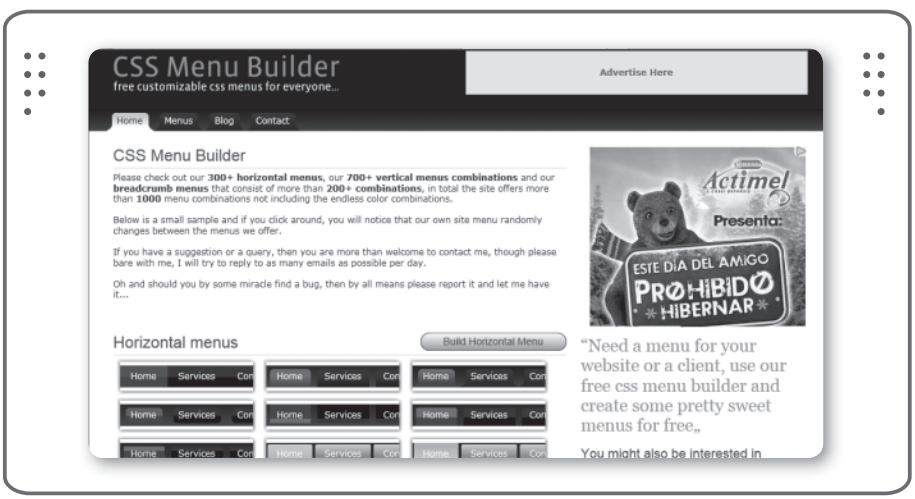
Con las herramientas para webmasters que nos ofrece Bing (www.bing.com/toolbox/webmaster) podremos conocer muchos datos adicionales sobre nuestro sitio, entre ellos algunos relacionados con tráfico, búsquedas y otras estadísticas muy útiles.

CSS MENU MAKER ● http://cssmenumaker.com

CSS Menu Maker (http://cssmenumaker.com) es un sitio que nos permite generar menús basados en HTML y CSS con una gran facilidad y en pocos pasos, además nos ofrece excelentes posibilidades de personalización y resultados realmente sorprendentes.

CSS MENU BUILDER ● http://www.cssmenubuilder.com

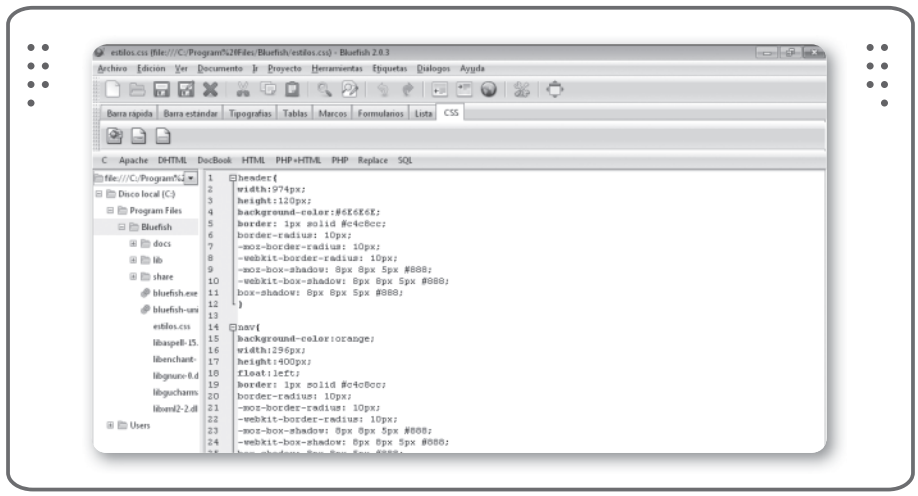
Con las alternativas que nos ofrece la aplicación online CSS Menu Builder (www.cssmenubuilder.com) podremos realizar la construcción de menús de navegación con una gran variedad de opciones, para ajustarlos luego a nuestras necesidades personales de diseño.



Programas relacionados

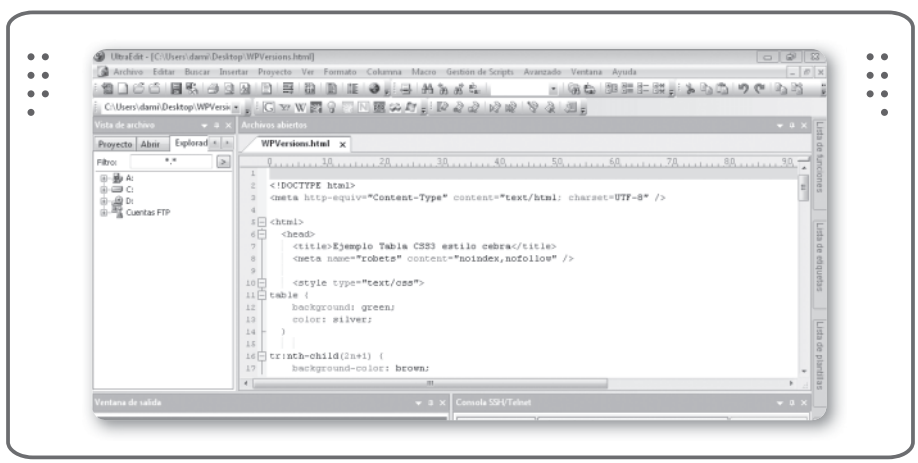
BLUEFISH ● http://bluefish.openoffice.nl/index.html

Bluefish (http://bluefish.openoffice.nl/index.html) es un editor de código de licencia GPL que ofrece muy buenas posibilidades para trabajar con lenguajes enfocados en el desarrollo web. Puede funcionar en sistemas Windows, Linux y también sobre Mac OSX.



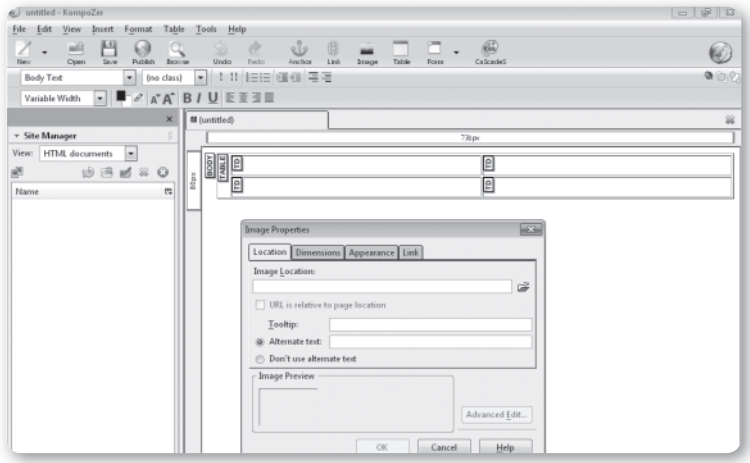
ULTRAEDIT ● http://www.ultraedit.com

UltraEdit se presenta como un potente editor de código con soporte para una gran variedad de lenguajes. Cuenta con versiones para Windows, Linux y Mac OSX. Es un producto pago, pero se puede conseguir una versión trial en el sitio www.ultraedit.com.



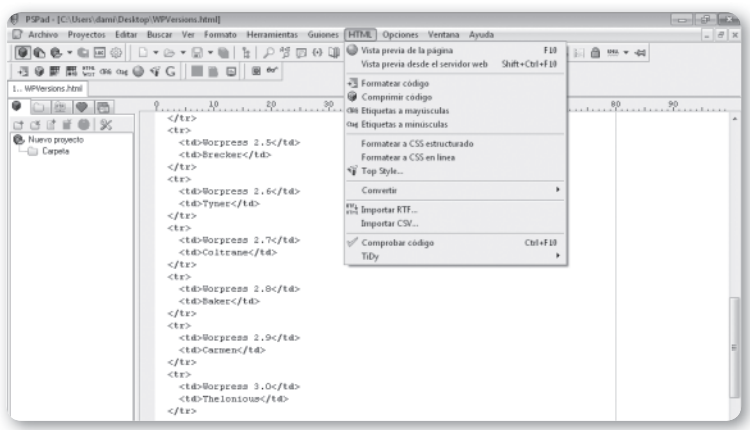
KOMPOZER ● http://kompozer.net

KompoZer (http://kompozer.net) es un editor de páginas web que ofrece las ventajas de una aplicación WYSIWYG. Entre sus características se destacan la versatilidad para edición de código y la posibilidad para gestionar archivos mediante FTP. Es software libre y multiplataforma.



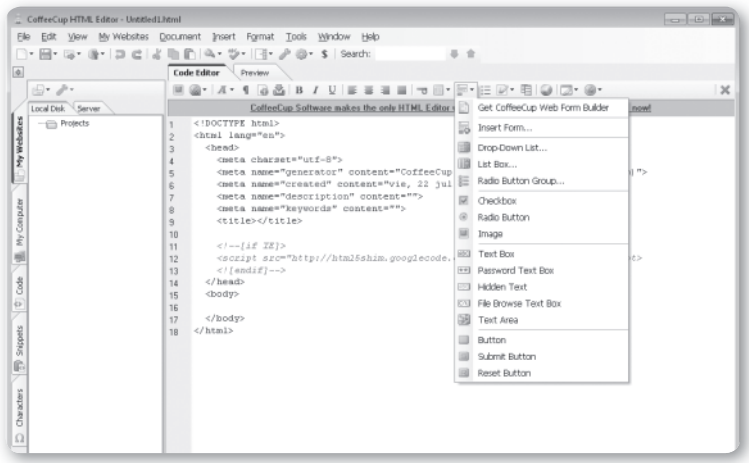
PSPAD ● http://www.pspad.com/es

PSPad es un editor de código para desarrolladores. Es capaz de identificar diversos lenguajes de programación web y permite resaltar su sintaxis con colores. Es freeware, funciona en Windows, está disponible en español y puede descargarse desde www.pspad.com/es.



COFFECUP HTML EDITOR ● http://www.coffecup.com/htmleditor

CoffeCup HTML Editor es una aplicación potente para crear páginas web. Ofrece menús que facilitan el acceso a las opciones y vistas de código para poder definir la estructura de nuestros documentos. Funciona en Windows y se puede obtener una versión trial ingresando en www.coffeecup.com/html-editor.

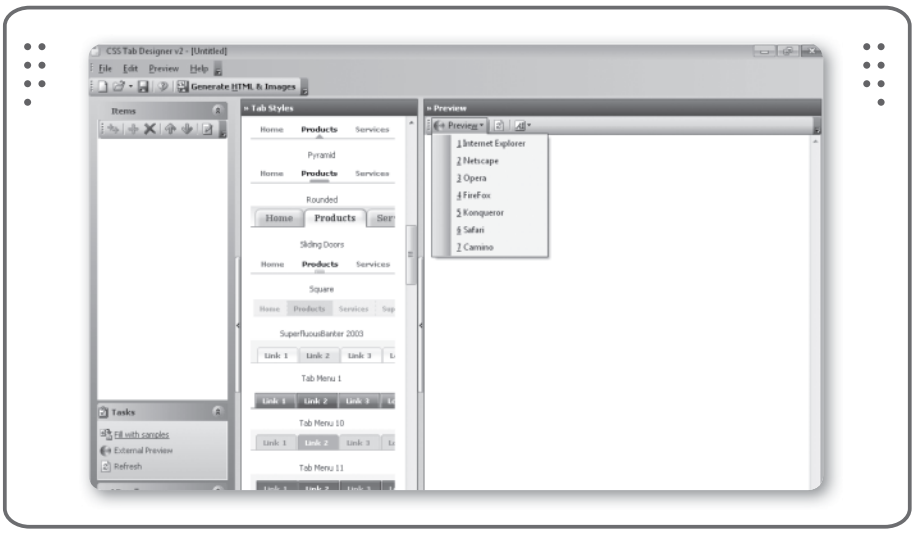


ARTISTEER ● http://www.artisteer.com

Artisteer es una aplicación que nos permite crear templates para nuestro sitio web o blog. Sin necesidad de escribir el código, podremos realizar plantillas compatibles con XHTML y CSS estándar, y también para CMSs como WordPress, Joomla o Druplal.

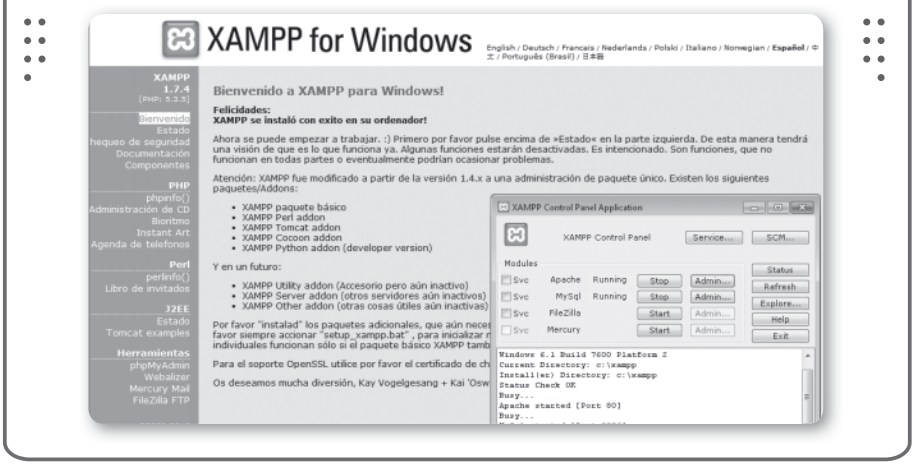
CSS TAB DESIGNER ● http://www.highdots.com/products

CSS Tab Designer es una herramienta freeware para Windows que nos permite crear menús y solapas basadas en CSS con gran facilidad. Podemos descargarlo visitando el sitio web que encontramos en la dirección www.highdots.com/products/css-tab-designer.



XAMPP ● http://www.apachefriends.org/es/xampp.html

XAMPP es un paquete (multiplataforma) que podremos instalar en nuestro equipo para tener en pocos pasos un completo servidor web Apache (con soporte paraMySQL y PHP). Se puede obtener ingresando en la dirección www.apachefriends.org/es/xampp.html.



CLAVES PARA COMPRAR
UN LIBRO DE COMPUTACIÓN

1 SOBRE EL AUTOR Y LA EDITORIAL

Revise que haya un cuadro "sobre el autor", en el que se informe sobre su experiencia en el tema. En cuanto a la editorial, es conveniente que sea especializada en computación.

2 PRESTE ATENCIÓN AL DISEÑO

Compruebe que el libro tenga guías visuales, explicaciones paso a paso, recuadros con información adicional y gran cantidad de pantallas. Su lectura será más ágil y atractiva que la de un libro de puro texto.

3 COMPARE PRECIOS

Suele haber grandes diferencias de precio entre libros del mismo tema; si no tiene el valor en tapa, pregunte y compare.

4 ¿TIENE VALORES AGREGADOS?

Desde un sitio exclusivo en la Red, un Servicio de Atención al Lector, la posibilidad de leer el sumario en la Web para evaluar con tranquilidad la compra, y hasta la presencia de adecuados índices temáticos, todo suma al valor de un buen libro.

5 VERIFIQUE EL IDIOMA

No solo el del texto; también revise que las pantallas incluidas en el libro estén en el mismo idioma del programa que usted utiliza.

usershop.redusers.com
VISITE NUESTRO SITIO WEB

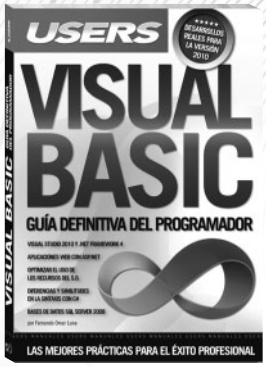
» Vea información más detallada sobre cada libro de este catálogo.
» Obtenga un capítulo gratuito para evaluar la posible compra de un ejemplar.
» Conozca qué opinaron otros lectores.
» Compre los libros sin moverse de su casa y con importantes descuentos.
» Publique su comentario sobre el libro que leyó.
» Manténgase informado acerca de las últimas novedades y los próximos lanzamientos.

TAMBIÉN PUEDE CONSEGUIR NUESTROS LIBROS EN KIOSCOS O PUESTOS DE PERIÓDICOS, LIBRERÍAS, CADENAS COMERCIALES, SUPERMERCADOS Y CASAS DE COMPUTACIÓN.

LLEGAMOS A TODO EL MUNDO VÍA OCA * Y DHL **

* SOLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com // usershop@redusers.com



Visual Basic

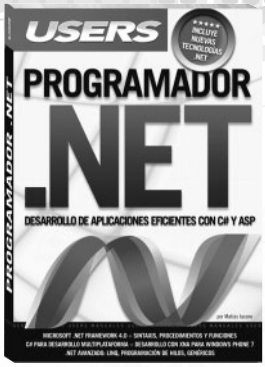
Este libro está escrito para aquellos usuarios que quieran aprender a programar en VB.NET. Desde el IDE de programación hasta el desarrollo de aplicaciones del mundo real en la versión 2010 de Visual Studio, todo está contemplado para conocer en profundidad VB.NET al finalizar la lectura.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-57-2

Microcontroladores

Este manual es ideal para aquellos que quieran iniciarse en la programación de microcontroladores. A través de esta obra, podrán conocer los fundamentos de los sistemas digitales, aprender sobre los microcontroladores PIC 16F y 18F, hasta llegar a conectar los dispositivos de forma inalámbrica, entre muchos otros proyectos.

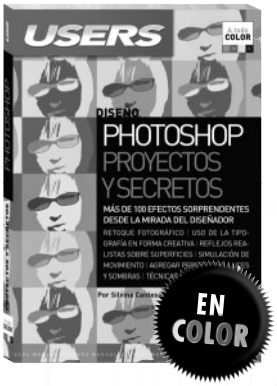
→ COLECCIÓN: MANUALES USERS
→ 320 páginas / 978-987-1773-56-5



Programador.NET

Este libro está dirigido a todos aquellos que quieran iniciarse en el desarrollo bajo lenguajes Microsoft. A través de los capítulos del manual, aprenderemos sobre POO y la programación con tecnologías .NET, su aplicación, cómo interactúan entre sí y de qué manera se desenvuelven con otras tecnologías existentes.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-26-8



Photoshop: proyectos y secretos

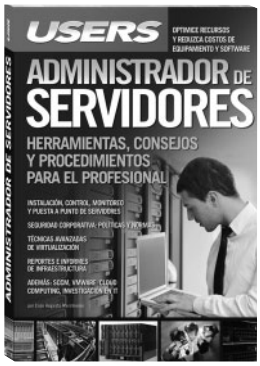
En esta obra aprenderemos a utilizar Photoshop, desde la original mirada de la autora. Con el foco puesto en la comunicación visual, a lo largo del libro adquiriremos conocimientos teóricos, al mismo tiempo que avanzaremos sobre la práctica, con todos los efectos y herramientas que ofrece el programa.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / 978-987-1773-25-1

WordPress

Este manual está dirigido a todos aquellos que quieran presentar sus contenidos o los de sus clientes a través de WordPress. En sus páginas el autor nos enseñará desde cómo llevar adelante la administración del blog hasta las posibilidades de interacción con las redes sociales.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-18-3



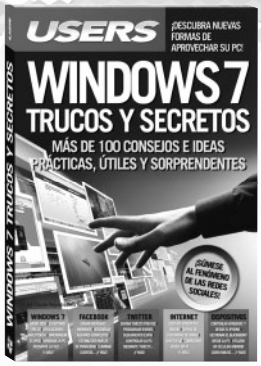
Administrador de servidores

Este libro es la puerta de acceso para ingresar en el apasionante mundo de los servidores. Aprenderemos desde los primeros pasos sobre la instalación, configuración, seguridad y virtualización; todo para cumplir el objetivo final de tener el control de los servidores en la palma de nuestras manos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-19-0

¡Léalo antes Gratis!

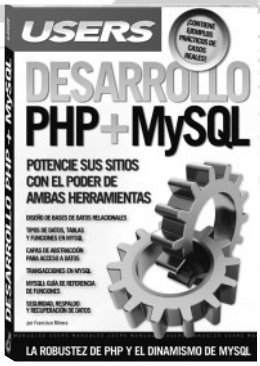
En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



Windows 7: Trucos y secretos

Este libro está dirigido a todos aquellos que quieran sacar el máximo provecho de Windows 7, las redes sociales y los dispositivos ultraportátiles del momento. A lo largo de sus páginas, el lector podrá adentrarse en estas tecnologías mediante trucos inéditos y consejos asombrosos.

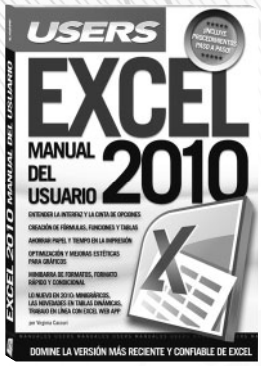
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-17-6



Desarrollo PHP + MySQL

Este libro presenta la fusión de dos de las herramientas más populares para el desarrollo de aplicaciones web de la actualidad: PHP y MySQL. En sus páginas, el autor nos enseñará las funciones del lenguaje, de modo de tener un acercamiento progresivo, y aplicar lo aprendido en nuestros propios desarrollos.

→ COLECCIÓN: MANUALES USERS
→ 432 páginas / ISBN 978-987-1773-16-9



Excel 2010

Este manual resulta ideal para quienes se inician en el uso de Excel, así como también para los usuarios que quieran conocer las nuevas herramientas que ofrece la versión 2010. La autora nos enseñará desde cómo ingresar y proteger datos hasta la forma de imprimir ahorrando papel y tiempo.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-15-2

Técnico Hardware

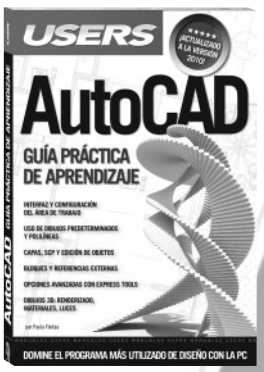
Esta obra es fundamental para ganar autonomía al momento de reparar la PC. Aprenderemos a diagnosticar y solucionar las fallas, así como a prevenirlas a través del mantenimiento adecuado, todo explicado en un lenguaje práctico y sencillo.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-14-5

PHP Avanzado

Este libro brinda todas las herramientas necesarias para acercar al trabajo diario del desarrollador los avances más importantes incorporados en PHP 6. En sus páginas, repasaremos todas las técnicas actuales para potenciar el desarrollo de sitios web.

→ COLECCIÓN: MANUALES USERS
→ 400 páginas / ISBN 978-987-1773-07-7



AutoCAD

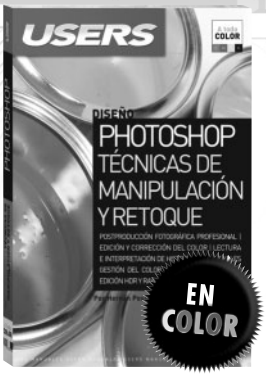
Este manual nos presenta un recorrido exhaustivo por el programa más difundido en dibujo asistido por computadora a nivel mundial, en su versión 2010. En sus páginas, aprenderemos desde cómo trabajar con dibujos predeterminados hasta la realización de objetos 3D.

→ COLECCIÓN: MANUALES USERS
→ 384 páginas / ISBN 978-987-1773-06-0

Windows 7 Avanzado

Esta obra nos presenta un recorrido exhaustivo que nos permitirá acceder a un nuevo nivel de complejidad en el uso de Windows 7. Todas las herramientas son desarrolladas con el objetivo de acompañar al lector en el camino para ser un usuario experto.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-08-4



Photoshop

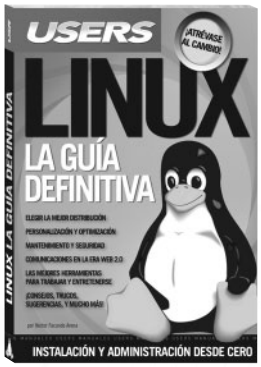
En este libro aprenderemos sobre las más novedosas técnicas de edición de imágenes en Photoshop. El autor nos presenta de manera clara y práctica todos los conceptos necesarios, desde la captura digital hasta las más avanzadas técnicas de retoque.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-05-3

Grabación y producción de música

En este libro repasaremos todos los aspectos del complejo mundo de la producción musical. Desde las cuestiones para tener en cuenta al momento de la composición, hasta la mezcla y el masterizado, así como la distribución final del producto.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-04-6



Linux

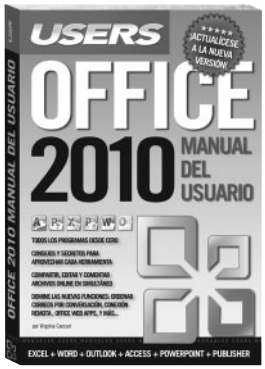
Este libro es una completa guía para migrar e iniciarse en el fascinante mundo del software libre. En su interior, el lector conocerá las características de Linux, desde su instalación hasta las opciones de entretenimiento, con todas las ventajas de seguridad que ofrece el sistema.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-8-6

Premiere + After Effects

Esta obra nos presenta un recorrido detallado por las aplicaciones audiovisuales de Adobe: Premiere Pro, After Effects y Soundbooth. Todas las técnicas de los profesionales, desde la captura de video hasta la creación de efectos, explicadas de forma teórica y práctica.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-9-3



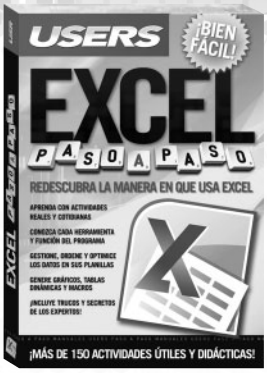
Office 2010

En este libro aprenderemos a utilizar todas las aplicaciones de la suite, en su versión 2010. Además, su autora nos mostrará las novedades más importantes, desde los minigráficos de Excel hasta Office Web Apps, todo presentado en un libro único.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-26013-6-2

¡Léalo antes Gratis!

En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



Excel Paso a Paso

En esta obra encontraremos una increíble selección de proyectos pensada para aprender, mediante la práctica, la forma de agilizar todas las tareas diarias. Todas las actividades son desarrolladas en procedimientos paso a paso de una manera didáctica y fácil de comprender.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-4-8

C#

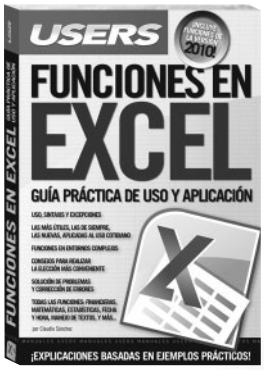
Este libro es un completo curso de programación con C# actualizado a la versión 4.0. Ideal tanto para quienes desean migrar a este potente lenguaje, como para quienes quieran aprender a programar desde cero en Visual Studio 2010.

→ COLECCIÓN: MANUALES USERS
→ 400 páginas / ISBN 978-987-26013-5-5

200 Respuestas: Seguridad

Esta obra es una guía básica que responde, en forma visual y práctica, a todas las preguntas que necesitamos contestar para conseguir un equipo seguro. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

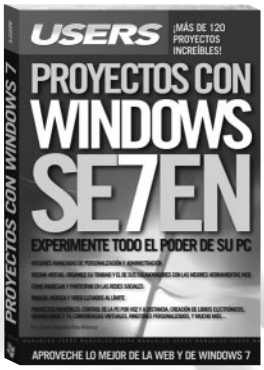
→ COLECCIÓN: 200 RESPUESTAS
→ 320 páginas / ISBN 978-987-26013-1-7



Funciones en Excel

Este libro es una guía práctica de uso y aplicación de todas las funciones de la planilla de cálculo de Microsoft. Desde las funciones de siempre hasta las más complejas, todas presentadas a través de ejemplos prácticos y reales.

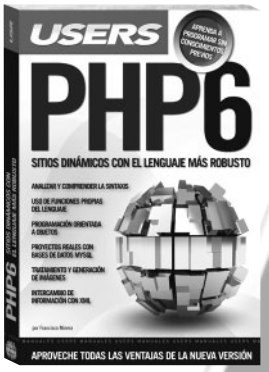
→ COLECCIÓN: MANUALES USERS
→ 368 páginas / ISBN 978-987-26013-0-0



Proyectos con Windows 7

En esta obra aprenderemos cómo aprovechar al máximo todas las ventajas que ofrece la PC. Desde cómo participar en las redes sociales hasta las formas de montar una oficina virtual, todo presentado en 120 proyectos únicos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-036-8



PHP 6

Este libro es un completo curso de programación en PHP en su versión 6.0. Un lenguaje que se destaca tanto por su versatilidad como por el respaldo de una amplia comunidad de desarrolladores, que lo convierten en un punto de partida ideal para quienes comienzan a programar.

→ COLECCIÓN: MANUALES USERS
→ 368 páginas / ISBN 978-987-663-039-9

USERS PRESENTA...

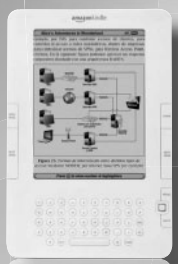
¡EL PRIMER EBOOK USERS!

Sí, ya podés leer Hackers al descubierto en tu PC, notebook, Amazon Kindle, iPad, en el celular...

CONSEGUILO DESDE CUALQUIER PARTE DEL MUNDO

A UN PRECIO INCREÍBLE

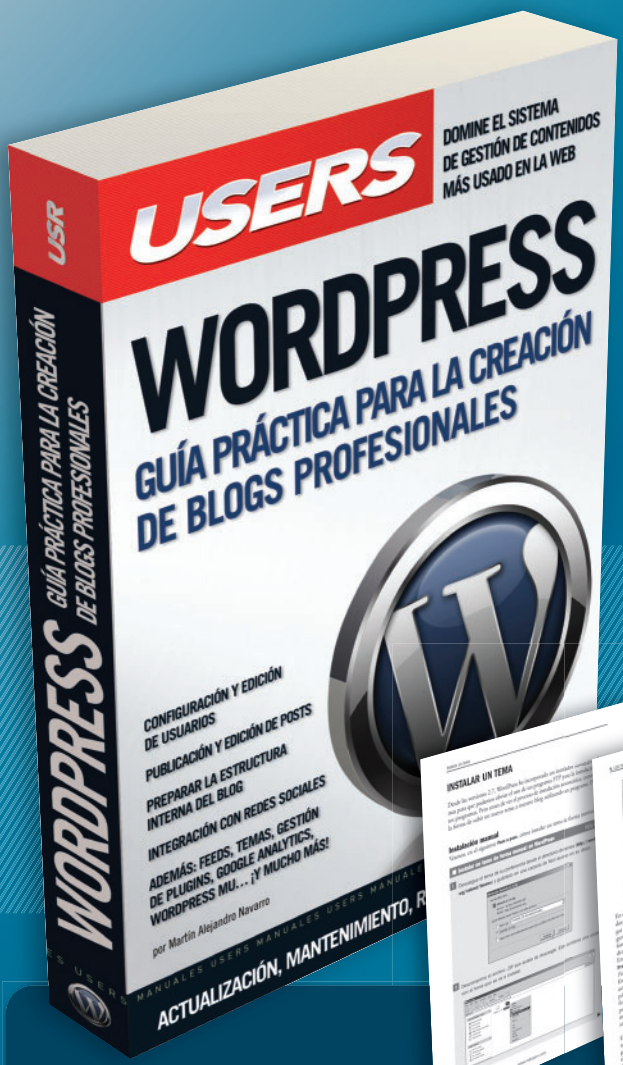
¿QUÉ ESTÁS ESPERANDO?



¡LEELO DONDE QUIERAS!

INGRESA YA A USERSHOP.REDUSERS.COM Y ENTERATE MÁS

GUÍA PRÁCTICA PARA LA CREACIÓN DE BLOGS PROFESIONALES



Este manual está dirigido a todos aquellos que quieran presentar sus contenidos o los de sus clientes a través de WordPress. En sus páginas el autor nos enseñará desde cómo llevar adelante la administración del blog hasta las posibilidades de interacción con las redes sociales.

» HOME / INTERNET
» 352 PÁGINAS
» ISBN 978-987-1773-18-3

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com // usershop@redusers.com

USERS

CONTENIDO

1 | INTRODUCCIÓN A TECNOLOGÍAS Y ESTÁNDARES WEB
Los orígenes de la Web / Web 2.0 / Web 3.0 / W3C / Lenguajes de etiquetas / Navegadores / Herramientas de desarrollo / Test de compatibilidad

2 | NOVEDADES DE HTML5
Elementos HTML4/XHTML / Elementos que se retiran del estándar o cambian con HTML5 / Características que incorpora HTML5

3 | ESTRUCTURA Y SEMÁNTICA
Semántica / Declaración de página y cabecera del documento / Estructura semántica de documentos en HTML5 / Estructurar una página en HTML5 / Microdatos / SEO

4 | TRABAJO CON ESTILOS
Interacción de HTML5 con los estilos CSS / Selectores, clases y propiedades / Construcción de reglas en CSS / Nuevas características de CSS3

5 | MULTIMEDIA
Audio con HTML5 / Video con HTML5 / La etiqueta <source> / Aspectos de compatibilidad de audio y video / Incorporar los elementos multimedia utilizando HTML5 en nuestro desarrollo / Acceso a dispositivos multimedia

6 | FORMULARIOS
Formularios HTML/XHTML / JavaScript y AJAX para potenciar los formularios / Características de formulario que incorpora HTML5 / Incorporar características HTML5 en un formulario

7 | DISEÑO Y DESARROLLO WEB ADAPTADO A MÓVILES
Ventajas de HTML5 / Cómo saber si el móvil está preparado para HTML5 / Técnicas de detección / Herramientas de desarrollo

8 | CARACTERÍSTICAS AVANZADAS
JavaScript y DOM / Canvas / SVG / WebGL (3D) / WebSockets / Geolocalización / Drag & Drop / WebStorage / Web Messaging / Server-Sent Events / Web Workers / Application cache API

NIVEL DE USUARIO

PRINCIPIANTE | INTERMEDIO | AVANZADO | EXPERTO

HTML5

En esta obra presentamos un nuevo paradigma de Internet que cambia de manera sustancial todo lo que conocíamos sobre el diseño y desarrollo web. Desde su concepción, está dirigida tanto a diseñadores como a desarrolladores web que buscan conocer las bases de HTML5 y CSS3.

A lo largo de sus capítulos, aprenderemos sobre los estándares web que existen, y conoceremos las diferencias entre las versiones anteriores y la actual del lenguaje.

Entre las mejoras que llegan de la mano de HTML5 se destacan las novedades en semántica, multimedia, formularios y el acceso a APIs, para crear la nueva generación de aplicaciones que revolucionan la Web.

Al finalizar la lectura, podremos realizar desarrollos modernos y profesionales, y conocer en profundidad HTML5 y CSS3 para comenzar a aplicarlos en nuestros proyectos.

RedUSERS.com

En este sitio encontrará una gran variedad de recursos y software relacionado, que le servirán como complemento al contenido del libro. Además, tendrá la posibilidad de estar en contacto con los editores, y de participar del foro de lectores, en donde podrá intercambiar opiniones y experiencias.

Si desea más información sobre el libro puede comunicarse con nuestro Servicio de Atención al Lector: usershop@redusers.com

HTML 5

We proudly introduce a new revolution in Web design and development: HTML5. In this book, the reader will learn the new aspects of CSS3 as well as HTML5 semantics, multimedia, forms, among other possibilities to achieve a richer Web development.

ISBN 978-987-1773-79-4
9 789871 773794

MANUALES USERS MANUALES USERS MANUALES USERS MANUALES USERS MAN

APLIQUE EL NUEVO ESTÁNDAR A SUS PROYECTOS